TUGAS AKHIR – TE 141599

# PENGEMBANGAN OSD (*ON SCREEN DISPLAY*) DENGAN PENAMBAHAN MENU UNTUK APLIKASI PADA *SEMI AUTONOMOUS MOBILE ROBOT* DENGAN LENGAN UNTUK MENGAMBIL OBJEK

Muhammad Saiful Hak
NRP 2214105082

Dosen Pembimbing
Ronny Mardiyanto, ST., MT., Ph.D.
Suwito, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

*FINAL PROJECT* – TE 141599

# *THE DEVELOPMENT OF OSD (ON SCREEN DISPLAY) WITH ADDITIONAL MENU FOR APPLICATION IN SEMI AUTONOMOUS MOBILE ROBOT WITH ARM TO PICK OBJECT*

Muhammad Saiful Hak
NRP 2214105082

*Supervisor*
Ronny Mardiyanto, ST., MT., Ph.D.
Suwito, ST., MT.

*ELECTRICAL ENGINEERING DEPARTMENT*
*Faculty of Industrial Technology*
*Sepuluh Nopember Institute of Technology*
Surabaya 2016

# PENGEMBANGAN OSD (*ON SCREEN DISPLAY*) DENGAN PENAMBAHAN MENU UNTUK APLIKASI PADA *SEMI AUTONOMOUS MOBILE ROBOT* DENGAN LENGAN UNTUK MENGAMBIL OBJEK

## TUGAS AKHIR

**Diajukan Guna Memenuhi Sebagian Persyaratan**
**Untuk Memperoleh Gelar Sarjana Teknik**
**Pada**
**Bidang Studi Elektronika**
**Jurusan Teknik Elektro**
**Fakultas Teknologi Industri**
**Institut Teknologi Sepuluh Nopember**

**Menyetujui:**

| Dosen Pembimbing I, | Dosen Pembimbing II, |
|---|---|
| Ronny Mardiyanto, ST., MT., Ph.D. | Suwito, ST., MT. |
| NIP: 198101182003121003 | NIP: 198101052005011004 |

**SURABAYA**
**JUNI, 2016**

**Pengembangan OSD (*On Screen Display*) dengan Penambahan Menu untuk Aplikasi pada *Semi Autonomous Mobile Robot* dengan Lengan untuk Mengambil Objek**

**Nama** **:** Muhammad Saiful Hak
**Pembimbing** **:** Ronny Mardiyanto, ST., MT., Ph.D.
Suwito, ST., MT.

# ABSTRAK

Tugas Akhir ini membahas pengembangan OSD (*On Screen Display*) dengan sistem FPV (*First Person View*) yang bertujuan untuk mendukung kinerja sebuah RC (*Remote Control*) atau *mobile robot* dengan cara mengambil data pada RC atau *mobile robot* tersebut. Penggunaan sensor pada modul OSD ini terbatas, sehingga dilakukan sebuah pengembangan dengan penambahan pengkabelan dan pemrograman menggunakan kompiler Arduino.

Dalam sistem ini, OSD yang dipakai adalah dua buah minimOSD yang telah dirancang dengan penambahan beberapa fitur menu sensor. Pada minimOSD yang pertama data GPS (*Global Positioning System*) berupa posisi lintang dan bujur didapat dari APM (*ArduPilot* Mega) yang diprogram menggunakan minimOSD *extra*. Pada minimOSD yang kedua didapat data sensor ultrasonik dan posisi lengan robot yang dikirim oleh Arduino Mega pada *mobile robot* menggunakan pengiriman serial. Sensor suhu, level baterai dan data waktu diperoleh dari penambahan pengkabelan pada pin-pin ATMega328. Hasil olah data sensor pada minimOSD pertama dan kedua yang berupa data visual digabung dan dikirim menuju layar monitor FPV menggunakan *video transmiter*. Animasi data yang ditampilkan mempunyai batas sebesar 256 data karakter, sehingga posisi lengan robot dan sensor ultrasonik hanya bisa digambarkan berupa perbandingan skala tingkatan dan data teks, sedangkan data sensor yang lain ditampilkan sesuai hasil olah data sebenarnya.

**Kata Kunci** – OSD (*On Screen Display*), FPV (*First Person View*), minimosd, *remote control*, APM (*ArduPilot Mega*), Arduino pro mini.

***The Development of OSD (On Screen Display) with Additional Menu for Application in Semi Autonomous Mobile Robot with Arm to Pick Object***

***Name*** **:** Muhammad Saiful Hak
***Supervisor*** **:** Ronny Mardiyanto, ST., MT., Ph.D.
Suwito, ST., MT.

## *ABSTRACT*

*This Final Project discusses the development of OSD (On Screen Display) with FPV (First Person View) system which aims to support the performance of an RC (Remote Control) or a mobile robot by taking the data from it. The use of sensors in the OSD module is limited, thus necessiating a development by additional wirings and programs made using the Arduino compiler.*

*In this system, the OSD used was two minimOSDs which was designed with the addition of several features of the sensor menu. In the first minimOSD, GPS (Global Positioning System) data in the form of longitude and latitude were obtained from APM (ArduPilot Mega) which were programmed using minimOSD extra. The second minimOSD collected the ultrasonic sensor data and the robot arm position which were sent by Arduino Mega on the mobile robot using serial communication. The temperature sensor, battery level and time data were obtained from the additional wirings at the ATmega328 pins. Results from the first and second minimOSD were combined and sent to the FPV monitor screen using video transmitter in the form of visual data. Animated data was limited to 256 characters, therefore the robot arm position and ultrasonic sensor data can only be shown as scale level comparions and text, while data from the other sensors were displayed according to the actual results.*

***Keywords*** *– OSD (On Screen Display), FPV (First Person View), minimosd, remote control, APM (ArduPilot Mega), Arduino pro mini.*

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Pemanfaatan media informasi saat ini semakin berkembang pesat dan tak lepas dari kehidupan manusia. Umumnya informasi digunakan oleh setiap orang untuk mengetahui sebuah atau kumpulan pesan tentang peristiwa tertentu yang diterima melalui proses komunikasi atau juga bisa didapatkan dari berita. Setiap orang pasti sangat memerlukan sebuah informasi, baik dari segi manapun. Karena dengan adanya informasi, seseorang dapat mengetahui sesuatu yang dibutuhkan atau tidak, dan juga bisa digunakan sebagai pembelajaran, pengalaman, atau instruksi. Pengertian dari informasi sendiri yaitu pesan (ucapan atau ekspresi) atau kumpulan pesan yang terdiri dari order sekuens dari simbol, atau makna yang dapat ditafsirkan dari pesan atau kumpulan pesan yang dapat direkam dan ditransmisikan. Dalam bidang ilmu komputer, informasi adalah data yang disimpan, diproses, atau ditransmisikan yang didapatkan dari proses pembelajaran atau instruksi dari suatu sistem. Begitu juga di dunia teknologi, khususnya bidang ilmu elektronika, informasi sangat bermanfaat dalam proses kinerja suatu sistem. Pemanfaatan informasi ini digunakan untuk memperoleh data-data seperti sensor atau data visual yang nantinya dapat dimonitoring secara *real-time* atau bisa dikatakan sebagai *multi-modal feedback*. Contoh dari pengambilan informasi ini seperti pada pengambilan data sensor oleh Arduino atau bisa juga oleh APM (*ArduPilot Mega*) yang nantinya diterjemahkan dalam bentuk visual menggunakan OSD (*On Screen Display*) dan ditransmisikan menggunakan sistem FPV (*First Person View*).

FPV yang juga dikenal sebagai *Remote Person View* (RPV) adalah metode yang digunakan untuk sistem pengendalian suatu objek atau kendaraan RC (*remote control*) menggunakan video, dimana driver dapat seolah-olah mengendalikan RC tersebut dari dalam RC itu sendiri. Kendaraan RC tersebut dikemudikan jarak jauh dari sudut pandang orang pertama melalui kamera onboard, transmisinya secara nirkabel ke layar monitor FPV. Penggunaan teknologi FPV ini semakin berkembang, seperti pada pesawat UAV (*Unmanned Aerial Vehicle*), drone, tank tanpa awak, mobil RC dan sejenisnya.

Penggunaan FPV hanya sebatas menampilkan data visual yang ditangkap oleh kamera. Untuk mendukung kinerja sebuah RC atau *mobile robot*, maka digunakan modul OSD (*On Screen Display*) seabagai informasi data yang diambil oleh RC atau *mobile robot* tersebut. Informasi data yang diambil oleh sistem *mobile robot* ini akan diterjemahkan kedalam bentuk data visual pada OSD tersebut yang nantinya akan ditransmisikan bersama data visual kamera menuju layar monitor FPV. OSD sendiri merupakan beberapa fitur menu konfigurasi yang ditampilkan dalam bentuk data visual pada layar monitor yang dibangkitkan oleh karakter generator yang ada didalam setiap chip mikrokontrol yang merupakan gudang pembangkit macam-macam huruf karakter serta berbagai macam bentuk gambar grafis.

Penggunaan OSD dengan sistem FPV pada tugas akhir ini akan membahas tentang perancangan dan pengembangan OSD dengan penambahan beberapa fitur menu seperti data sensor Ultrasonik, data temperatur, posisi lengan robot, level *battery* dan RTC (*Real Time Clock*) yang diaplikasikan pada *Semi Autonomous Mobile Robot* dengan lengan untuk pengambilan sebuah objek. OSD yang dipakai nantinya akan menggunakan minimOSD yang diprogram dengan kontroler Arduino. Pada minimOSD ini dilakukan sebuah penambahan *pin wiring* untuk proses pembacaan sensor temperatur, RTC dan level *battery*. Selain itu, untuk beberapa sensor seperti sensor Ultrasonik dan lengan robot dikonfigurasikan dengan Arduino yang ada pada *mobile robot* tersebut. Untuk data GPS (*Global Positioning System*) sendiri, menggunakan modul APM (*ArduPilot Mega*) yang nantinya akan dikombinasikan dengan data-data sensor *extra wires* pada minimosd dan ditransmisikan bersama data visual kamera menggunakan sistem FPV menuju layar monitor FPV.

## 1.2 Rumusan Masalah

Sehubungan dengan latar belakang yang telah dijelaskan sebelumnya, terdapat beberapa masalah yang akan dibahas antara lain sebagai berikut :

1. Bagaimana mengolah data dan memrogram minimosd menggunakan Arduino.
2. Bagaimana menambahkan beberapa fitur menu atau data-data sensor yang terdapat pada *Semi Autonomous Mobile Robot* dengan *extra wires* pada minimosd.

3. Bagaimana menganimasikan jarak sensor Ultrasonik dan posisi lengan robot pada layar monitor FPV.
4. Bagaimana mengkombinasikan data sensor *extra wires*, data GPS dengan APM, dan data visual kamera untuk dikirm menggunakan sistem FPV.

## 1.3 Batasan Masalah

Batasan masalah yang akan dibahas dalam Tugas Akhir ini adalah :

1. Memprogram minimosd menggunakan kontroler Arduino tanpa chip dan dengan compiler Arduino.
2. Perancangan *extra wiring* pada minimosd untuk penambahan sensor temperatur, RTC, dan level *battery*.
3. Menerima dan mengolah data sensor ultrasonik dan posisi lengan robot dari Arduino Mega yang terdapat pada *Semi Autonomous Mobile Robot*.
4. Animasi jarak sensor ultrasonik dan posisi lengan robot hanya berupa level perbandingan skala.
5. Menggunakan dua minimosd untuk kombinasi data-data sensor *extra wires* dan data GPS dengan APM yang diprogram dengan minimosd extra.

## 1.4 Tujuan

Tujuan dari tugas akhir ini adalah merancang dan merealisasikan pengembangan OSD dengan penambahan beberapa fitur menu berupa *extra wires* sebagai pendukung aplikasi pada *Semi Autonomous Mobile Robot*. Dengan menggunakan OSD dapat mempermudah penggunanya mengetahui data-data yang diambil oleh *mobile robot* tersebut, seperti data sensor ultrasonik, temperatur, RTC, GPS, level *battery*, dan posisi lengan *robot* yang dikirim dan ditampilkan pada layar monitor menggunakan sistem FPV.

## 1.5 Metodologi

Metodologi yang digunakan dalam penyusunan tugas akhir ini sebagai berikut :

1. Studi literatur

   Studi literatur tugas akhir ini bersumber pada jurnal-jurnal, buku referensi dan *datasheet* komponen yang digunakan. Selain itu,

mengidentifikasi masalah yang akan dibahas dan mempelajari buku pedoman dari beberapa sumber yang menunjang teori atau prinsip dasar rangkaian, baik *software* maupun *hardware* dalam pembuatan sistem tersebut. Dalam kegiatan ini juga dilakukan beberapa survei lapangan atau analisa kondisi lingkungan, guna mengetahui seberapa besar sistem ini mempunyai nilai kebermanfaatan jika benar-benar direalisasikan.

2. Perancangan dan realisasi alat
   Perancangan alat pada tugas akhir ini dimulai dari pemrograman dan perancangan tampilan beberapa fitur menu pada OSD menggunakan kontroler Arduino, pengambilan data-data sensor oleh Arduino pada *mobile robot* dan juga oleh minimosd itu sendiri yang setelahnya dikonversi kedalam bentuk visual, *wiring* minimOSD menggunakan sistem FPV, pengiriman data video oleh *transmitter* menuju layar monitor FPV.
3. Tahap pengujian sistem dan analisa
   Pengujian alat dimulai dengan terlebih dahulu menguji respon tiap komponen. Apabila respon tiap komponen sudah sesuai, maka dilakukan penggabungan dan uji fungsi alat. Dari pengujian ini juga dilakukan penganalisaan sistem, untuk memperoleh data-data sistem, seberapa besar nilai error pada sistem.
4. Pembuatan laporan
   Penyusunan laporan dilakukan mengacu pada perancangan dan realisasi alat, serta pengujian alat, sehingga hasil yang diperoleh dari pembuatan alat dapat dijelaskan secara rinci dan spesifik sesuai dengan data-data yang diperoleh.

## 1.6 Sistematika Penulisan

Sistematika penulisan pada tugas akhir ini dibagi menjadi beberapa bab dengan rincian :

BAB I : PENDAHULUAN
Menguraikan latar belakang, perumusan masalah, batasan masalah, tujuan dan manfaat yang berkaitan dengan pengerjaan dan penyusunan Tugas Akhir ini.

BAB II : TEORI PENUNJANG
Pada bab ini dikemukakan berbagai macam dasar teori yang berhubungan dengan permasalahan yang dibahas, antara lain meliputi teori tentang sistem FPV, pemrograman minimosd, *wiring* minimosd, atmega328,

max7456, arduino, program minimosd extra, dan sensor-sensor yang digunakan.

BAB III : PERANCANGAN DAN REALISASI ALAT

Berisi tentang tahap-tahap perancangan *extra wires* minimosd, baik secara *hardware* dan *software*. Perancangan beberapa sensor-sensor dan kombinasi data sensor, serta perancangan tampilan visual untuk ditampilkan pada moniotr FPV.

BAB IV : PENGUJIAN DAN ANALISA SISTEM

Bab ini membahas mengenai pengujian dari sistem yang telah diimplementasikan pada *Semi Autonomous Mobile Robot* dan analisa data berdasarkan parameter yang ditetapkan.

BAB V : PENUTUP

Berisi tentang kesimpulan dan saran yang diperoleh dalam Tugas Akhir ini.

## 1.7 Relevansi

Manfaat dari tugas akhir ini adalah mengetahui kerja sistem *mobile robot* atau sebagai multi-modal *feedback*. Sistem ini untuk mempermudah penggunanya mengetahui data-data yang diambil oleh *mobile robot* tersebut, seperti data sensor ultrasonik, temperatur, RTC, GPS, level *battery*, dan posisi lengan *robot* yang dikirim dan ditampilkan pada layar monitor menggunakan sistem FPV.

Hasil yang dicapai diharapkan dapat menjadi salah satu referensi dalam pengembangan OSD menggunakan sistem FPV yang diaplikasi pada *mobile robot* atau RC (*Remote Control*).

*--Halaman ini sengaja dikosongkan--*

# BAB II
# TEORI PENUNJANG

## 2.1 Arduino

Arduino adalah pengendali mikro single-board yang bersifat *open-source*, diturunkan dari *wiring platform*, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. *Hardware*-nya memiliki prosesor *Atmel AVR* dan *software*-nya memiliki bahasa pemrograman sendiri. Arduino juga merupakan *platform hardware* terbuka yang ditujukan kepada siapa saja yang ingin membuat purwarupa peralatan elektronik interaktif berdasarkan *hardware* dan *software* yang fleksibel dan mudah digunakan. Mikrokontroler diprogram menggunakan bahasa pemrograman arduino yang memiliki kemiripan *syntax* dengan bahasa pemrograman C. Karena sifatnya yang terbuka maka siapa saja dapat mengunduh skema *hardware* arduino dan membangunnya. Arduino menggunakan keluarga mikrokontroler ATMega yang dirilis oleh Atmel sebagai basis, namun ada individu/perusahaan yang membuat *clone* arduino dengan menggunakan mikrokontroler lain dan tetap kompatibel dengan arduino pada level *hardware*. Untuk fleksibilitas, program dimasukkan melalui bootloader meskipun ada opsi untuk mem-*bypass bootloader* dan menggunakan *downloader* untuk memprogram mikrokontroler secara langsung melalui port ISP (*In System Programming*). [1]


**Gambar 2.1** Arduino Uno [1]

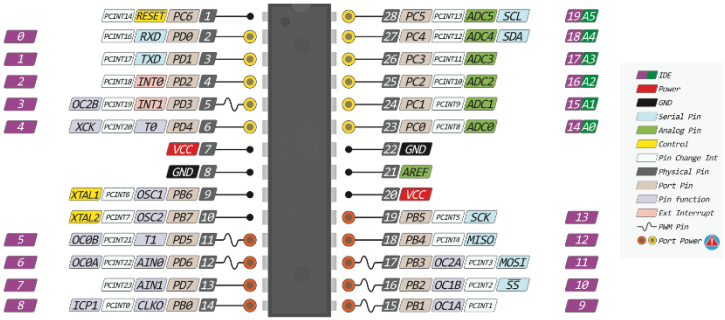
### 2.1.1 ATMega 328

ATMega328 adalah mikrokontroller keluaran dari atmel yang mempunyai arsitektur RISC (*Reduce Instruction Set Computer*) yang dimana setiap proses eksekusi data lebih cepat dari pada arsitektur CISC (Completed Instruction Set Computer).

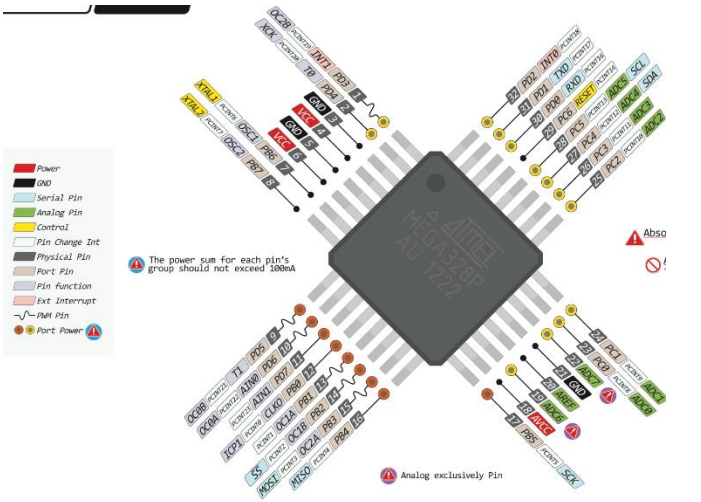
ATMega328 memiliki beberapa fitur antara lain:

1. 130 macam instruksi yang hampir semuanya dieksekusi dalam satu siklus *clock*.
2. 32 x 8-bit register serba guna.
3. Kecepatan mencapai 16 MIPS dengan *clock* 16 MHz.
4. 32 KB *Flash memory* dan pada arduino memiliki *bootloader* yang menggunakan 2 KB dari flash memori sebagai *bootloader*.
5. Memiliki EEPROM (*Electrically Erasable Programmable Read Only Memory*) sebesar 1KB sebagai tempat penyimpanan data semi *permanent* karena EEPROM tetap dapat menyimpan data meskipun catu daya dimatikan.
6. Memiliki SRAM (*Static Random Access Memory*) sebesar 2KB.
7. Memiliki pin I/O digital sebanyak 14 pin 6 diantaranya PWM (*Pulse Width Modulation*) output.
8. Master / Slave SPI *Serial interface.*

Mikrokontroller ATmega 328 memiliki arsitektur Harvard, yaitu memisahkan memori untuk kode program dan memori untuk data sehingga dapat memaksimalkan kerja dan *parallelism*. Instruksi - instruksi dalam memori program dieksekusi dalam satu alur tunggal, dimana pada saat satu instruksi dikerjakan instruksi berikutnya sudah diambil dari memori program. Konsep inilah yang memungkinkan instruksi – instruksi dapat dieksekusi dalam setiap satu siklus *clock*. 32 x 8-bit register serba guna digunakan untuk mendukung operasi pada ALU (*Arithmatic Logic Unit*) yang dapat dilakukan dalam satu siklus. 6 dari register serbaguna ini dapat digunakan sebagai 3 buah register pointer 16-bit pada mode pengalamatan tidak langsung untuk mengambil data pada ruang memori data. Ketiga register pointer 16-bit ini disebut dengan register X (gabungan R26 dan R27), register Y (gabungan R28 dan R29), dan register Z (gabungan R30 dan R31). Hampir semua instruksi AVR memiliki format 16-bit. Setiap alamat memori program terdiri dari instruksi 16-bit atau 32-bit. Selain register serba guna di atas, terdapat register lain yang terpetakan dengan teknik *memory mapped* I/O selebar 64 byte. Beberapa register ini digunakan untuk fungsi khusus antara lain sebagai register *control Timer*/*Counter*, Interupsi, ADC, USART, SPI, EEPROM, dan fungsi I/O lainnya. Register – register ini menempati memori pada alamat 0x20h – 0x5Fh. [2]




**Gambar 2.2** Pin *Mapping* ATMega 328




**Gambar 2.3** Pin *Mapping* ATMega 328 SMD (*Surface Mount Device*)

ATMega 328 memiliki 3 buah PORT utama yaitu PORTB, PORTC, dan PORTD dengan total pin *input/output* sebanyak 23 pin. PORT tersebut dapat difungsikan sebagai *input/output* digital atau difungsikan sebagai periperal lainnya [3].

1. PORTB

   *Port B* merupakan jalur data 8bit yang dapat difungsikan sebagai *input/output*. Selain itu PORTB juga dapat memiliki fungsi alternatif seperti berikut.

a. ICP1 (PB0), berfungsi sebagai *Timer Counter* 1 *input capture* pin.
b. OC1A (PB1), OC1B (PB2) dan OC2 (PB3) dapat difungsikan sebagai keluaran PWM (*Pulse Width Modulation*).
c. MOSI (PB3), MISO (PB4), SCK (PB5), SS (PB2) merupakan jalur komunikasi SPI.
d. Selain itu pin ini juga berfungsi sebagai jalur pemograman serial (ISP).
e. TOSC1 (PB6) dan TOSC2 (PB7) dapat difungsikan sebagai sumber *clock* external untuk *timer*.
f. XTAL1 (PB6) dan XTAL2 (PB7) merupakan sumber *clock* utama mikrokontroler.

2. PORTC

*Port C* merupakan jalur data 7bit yang dapat difungsikan sebagai *input/output* digital. Fungsi alternatif PORTC antara lain sebagai berikut.

a. ADC6 *channel* (PC0, PC1, PC2, PC3, PC4, PC5) dengan resolusi sebesar 10 bit. ADC dapat kita gunakan untuk mengubah input yang berupa tegangan analog menjadi data digital
b. I2C (SDA dan SDL) merupakan salah satu fitur yang terdapat pada PORTC. I2C digunakan untuk komunikasi dengan sensor atau *device* lain yang memiliki komunikasi data tipe I2C seperti sensor kompas, *accelerometer nunchuck*.

3. PORTD

*Port D* merupakan jalur data 8bit yang masing-masing pin-nya juga dapat difungsikan sebagai *input/output*. Sama seperti *PortB* dan *PortC*, *PortD* juga memiliki fungsi *alternative* dibawah ini.

a. USART (TXD dan RXD) merupakan jalur data komunikasi serial dengan level sinyal TTL. Pin TXD berfungsi untuk mengirimkan data serial, sedangkan RXD kebalikannya yaitu sebagai pin yang berfungsi untuk menerima data serial.

b. *Interrupt* (INT0 dan INT1) merupakan pin dengan fungsi khusus sebagai interupsi *hardware*. Interupsi biasanya digunakan sebagai selaan dari program, misalkan pada saat program berjalan kemudian terjadi interupsi *hardware/software* maka program utama akan berhenti dan akan menjalankan program interupsi.
c. XCK dapat difungsikan sebagai sumber *clock external* untuk USART, namun kita juga dapat memanfaatkan *clock* dari CPU, sehingga tidak perlu membutuhkan *external clock.*
d. T0 dan T1 berfungsi sebagai masukan *counter external* untuk *timer* 1 dan *timer* 0.
e. AIN0 dan AIN1 keduanya merupakan masukan *input* untuk *analog comparator*.

## 2.2 OSD (*On Screen Display*)

OSD singkatan dari "*On Screen Display*." Ini adalah menu konfigurasi termasuk layar dengan berbagai monitor. Ketika anda menekan tombol MENU, atau salah satu tombol di bagian depan atau samping monitor, di layar mungkin akan muncul. OSD biasanya menyertakan pilihan kalibrasi monitor, seperti kecerahan, kontras, dan warna. Beberapa OSD mungkin juga termasuk posisi horizontal dan vertikal kontrol serta penyesuaian *tilt* dan *keystone*. Jika anda tidak dapat membuat penyesuaian pada pengaturan monitor menggunakan interface OSD, maka tampilan layar dapat dikunci. Anda mungkin juga melihat frasa "OSD Dikunci" muncul ketika anda mencoba untuk mengubah pengaturan. Beberapa monitor termasuk salah satu pilihan dalam menu yang disebut "OSD Lock", anda dapat mengaktifkan atau menonaktifkan. Jika opsi ini tersedia, hanya gilirannya OSD Lock aktif dan anda akan dapat melakukan penyesuaian pengaturan. Jika monitor anda tidak memasukkan "OSD Lock" pilihan, anda mungkin dapat membuka OSD dengan menekan dan menahan MENU tombol di bagian depan atau samping monitor selama beberapa detik. Jika Anda mencoba metode ini, bersabarlah! Ini bisa memakan waktu hingga 15 detik untuk OSD untuk membuka. OSD dibangkitkan oleh sirkit karakter generator yang ada didalam setiap chip mikrokontrol yang merupakan gudang pembangkit macam-macam huruf karakter seperti yang ada pada tombol mesin ketik serta berbagai macam bentuk gambar grafis. Misalnya bentuk gambar grafis *volume*, kontras, *color*, *brightnes*s. Karakter

generator membutuhkan bantuan sinyal pulsa-pulsa seperti dibawah ini untuk dapat menampilkan OSD. [3]

- Pulsa *Horizontal Sync* (HS) yang diperoleh dari pulsa *flyback* (pin-AFC).
- Pulsa *Vertical Sync* (VS) yang diperoleh dari IC (*Integrated Circuit*) *Vertical-Out*.
- Osilator OSD - untuk membangkitkan pulsa-pulsa *clock* yang dibutuhkan karakter generator. Osilator OSD tidak menggunakan x-tal tetapi menggunakan sirkit LC (koil dan kapasitor). Pada televisi model baru osilator OSD sudah tidak dijumpai lagi, karena pulsa-pulsa *clock* untuk OSD dapat diturunkan dari osilator sistem.

Keluaran atau output karakter generator adalah :

- *Data OSD Red*
- *Data OSD Green*
- *Data OSD Blue*
- *Sinyal Fast Blank* yang merupakan sinyal yang digunakan untuk menggerakkan switch kecepatan tinggi yang ada di IC video-chroma. Didalam IC chroma antara gambar televisi dan data OSD sebenarnya ditampilkan secara bergantian dengan kecepatan tinggi, dengan tujuan agar tampilan OSD lebih jelas.

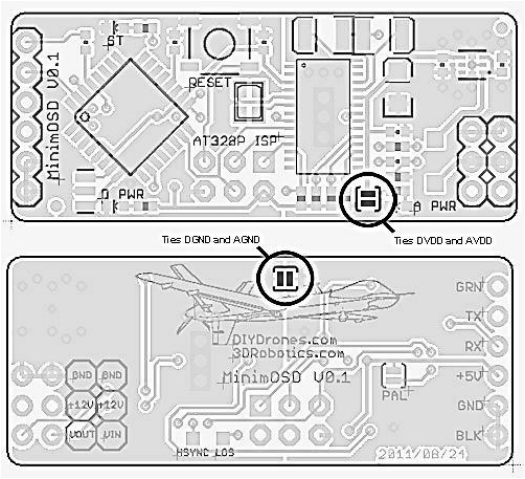
Pada gambar 2.4 merupakan contoh tampilan beberapa informasi data yang diproses oleh OSD pada layar monitor.


**Gambar 2.4** Tampilan oleh OSD pada Layar Monitor

### 2.2.1 MinimOSD

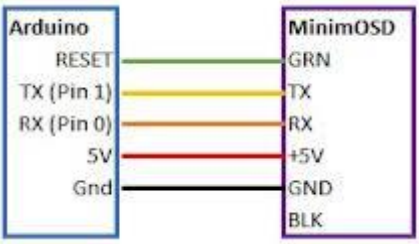
MinimOSD adalah papan super-kecil yang dirancang oleh 3D Robotika. Itu semua yang dibutuhkan untuk mendapatkan data telemetri OSD dari ArduPilot Mega. Hanya menghubungkan kamera FPV dan link video dan anda siap untuk terbang dengan instrumen di layar.

Didalam minimosd terdapat rangkaian dengan chip ATMega 328 dan MAX7456. ATMega 328 tersebut menerima data dari APM dengan komunikasi serial TX RX dan setelahnya akan dikonversi menjadi sebuah karakter oleh Max7456. Komunikasi yang digunakan antara ATMega 328 dengan Max7456 yaitu menggunakan komunikasi pengiriman SPI (*Serial Peripheral Interface*). [4]




**Gambar 2.5** MinimOSD [4]

MinimOSD biasanya digunakan pada beberapa robot tanpa awak seperti pada drone, pesawat UAV, atau pada *mobile robot*. Menggunakan minimOSD dikarenakan bentuk *hardware* yang kecil dan minimalis, tidak terlalu memakan ruang jika diaplikasikan pada mobile robot. Selain itu, minimOSD dapat diprogram menggunakan kontroler Arduino. Gambar rangkaian interface antara minimOSD dan Arduino dapat dilihat pada gambar 2.6. [4]




**Gambar 2.6** Interfacing Arduino-MinimOSD

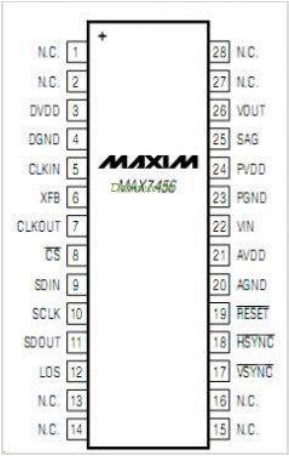
Untuk memprogram minimosd juga dapat menggunakan kabel FTDI (*Future Technology Devices International*). Kabel FTDI adalah

sebuah konverter dari USB to Serial TTL (*Transistor Transistor Logic*) Level.

### 2.2.2 MAX7456

Max7456 adalah salah satu IC (*Integrated Circuit*) dengan *single channel* OSD monokrom yang digunakan untuk driver video eksternal, sync separator, video switch, dan EEPROM. Semua pasar nasional maupun internasional menggunakan 256 data karakter pemrogram dalam mode standart NTSC dan PAL. Max7456 dengan mudah menampilkan beberapa informasi berupa data visual seperti logo, grafis kustom, waktu dan tanggal dengan karakter dan ukuran yang ditentukan. IC ini hanya memuat 256 karakter dan gambar grafis yang bisa diprogram menggunakan port SPITM. Terdapat 28 pin pada IC ini dan bertahan pada range suhu -40 derajat sampai dengan 85 derajat *Celcius.*

Sedangkan pada aplikasi minimosd disini sendiri, Max7456 dikonfigurasikan dengan ATMega 328 dan diprogram menggunakan kontroler Arduino. Berikut pada gambar 2.7 merupakan gambar dari IC Max7456. [5]

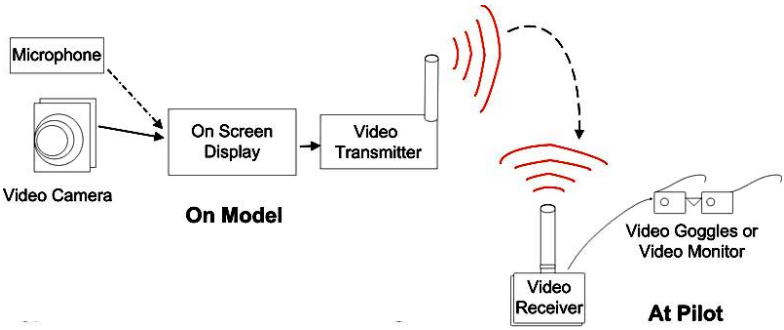



**Gambar 2.7** IC Max7456 [5]

## 2.3 FPV (*First Person View*)

FPV juga dikenal sebagai *Remote Person View* (RPV), atau *piloting* dengan video, adalah metode yang digunakan untuk mengontrol

kendaraan *remote control* dari driver atau pilot view point. Paling sering digunakan untuk pilot pesawat radio kontrol atau jenis lain dari kendaraan udara tak berawak (UAV). Kendaraan baik didorong atau dikemudikan jarak jauh dari sudut pandang orang pertama melalui kamera onboard, secara nirkabel ke kacamata video yang FPV atau monitor video. Pengaturan yang lebih canggih termasuk pan-dan-tilt kamera *gimbaled* dikendalikan oleh sensor giroskop di kacamata pilot dan dengan dual kamera terintegrasi, memungkinkan tampilan stereoscopic benar. [6]




**Gambar 2.8** Sistem FPV

## 2.4 APM (*ArduPilot Mega*)

Ardupilot Mega adalah salah satu open source autopilot hardware dan software dibidang aerial. Ardupilot Mega atau lebih dikenal dengan APM ini merupakan embeded system yang terintegrasi dengan IMU (*Inertia Measurement Unit*) dan GPS dengan menggunakan platform arduino mega (ATmega1280/2560).

APM ini sebuah open source platform kendaraan tanpa awak, mampu mengendalikan multicopters otonom, pesawat sayap tetap, helikopter tradisional dan kendaraan beroda. Ardupilot adalah platform pemenang penghargaan yang memenangkan kompetisi 2012 dan 2014 UAV Outback Challenge. Ini diciptakan pada tahun 2007 oleh komunitas DIY Drones. Hal ini didasarkan pada Arduino open-source elektronik platform yang prototyping. Versi Ardupilot pertama didasarkan pada thermopile, yang bergantung pada penentuan lokasi cakrawala relatif terhadap pesawat dengan mengukur perbedaan suhu antara langit dan tanah. [4]

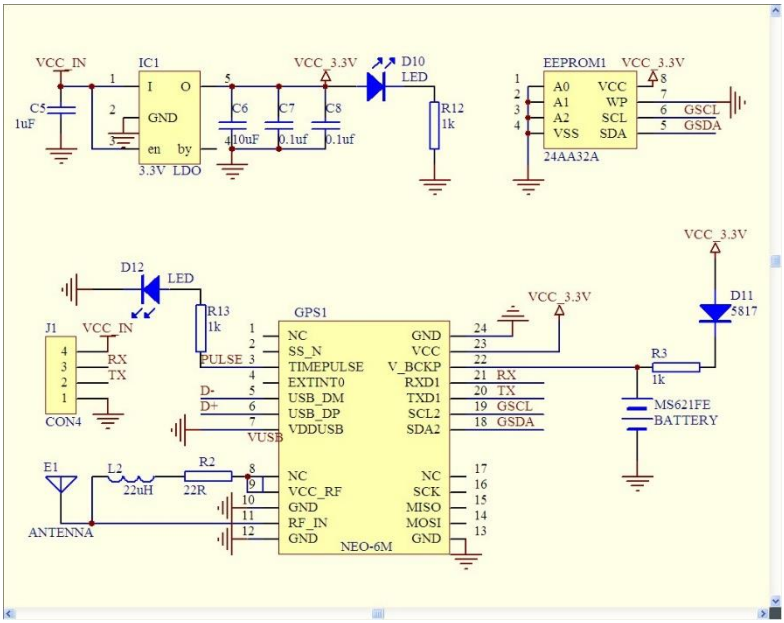
Kemudian, sistem diperbaiki untuk menggantikan thermopiles dengan *Inertial Measurement Unit* (IMU) menggunakan kombinasi accelerometers, giroskop dan magnetometer. Dan sekarang, proyek ArduPilot telah berkembang ke berbagai produk hardware dan software, termasuk APM dan garis Pixhawk/PX4 dari pilot otomatis, dan ArduCopter, ArduPlane serta ArduRover untuk proyek perangkat lunak. Pendekatan perangkat lunak bebas dari Ardupilot mirip dengan yang dari PX4 / Pixhawk dan Paparazzi Project, di mana biaya rendah dan ketersediaan memungkinkan penggunaan hobi dalam kecil pesawat jarak jauh dikemudikan, seperti kendaraan udara mikro dan UAV miniatur.

## 2.5 GPS (*Global Positioning System*)

*Global positioning system* atau disingkat GPS adalah suatu teknologi navigasi yang menyediakan informasi posisi dan waktu kapan saja, diamana saja atau dekat dengan bumi, dalam cuaca apapun selama tidak ada yang menghalagi pandangan empat atau lebih satelit GPS. GPS adalah teknologi yang dibuat dan dirawat oleh pemerintah Amerika Serikat serta menjadikannya sebagai sarana yang bebas di akses menggunakan penerima sinyal GPS. GPS sangat banyak digunakan dalam bidang militer, komersial, dan juga seluruh pengguna komersial diseluruh dunia. GPS adalah suatu teknologi yang akan menunjukkan posisi penerima sinyal GPS yang ada pada suatu benda.

Untuk mengetahui lokasi suatu benda, penerima sinyal GPS memerlukan dua data, yaitu data waktu dan posisi satelit GPS. Satelit GPS mempunyai jam atom, yaitu jam dengan akurasi yang sangat tinggi. GPS memancarkan sinyal data waktu data tersebut dikirim. Selisih waktu sinyal terkirim dan sampai dapat digunakan untuk menghitung jarak penerima sinyal GPS dengan satelit pengirim. Satelit GPS juga mengetahui posisi orbitnya. Dengan memperoleh data kapan sinyal dikirim satelit GPS dan diterima oleh penerima sinyal GPS, *receiver* GPS dapat mengetahui jarak dan posisi*receiver*dari satelit. Data dari 4 satelit dapat digunakan untuk mengetahui lokasi receiver secara 3D, yaitu lokasi *longitude* (lintang), *latitude* (bujur), dan *altitude* (ketinggian). [7]

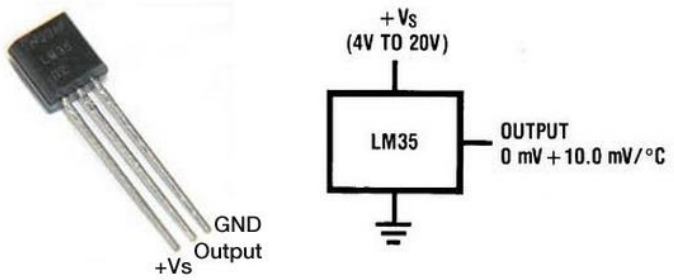
Penelitian ini menggunakan data GPS yang diolah Arduino untuk menentukan posisi Robot. GPS juga digunakan untuk mengetahui rute yang telah ditempuh robot dan juga merencanakan rute. GPS sangat berguna untuk melacak jejak dan posisi robot. GPS yang digunakan adalah 3DR GPS Ublox. Berikut ini adalah rangkaian 3DR GPS Ublox.




**Gambar 2.9** Rangkaian 3DR GPS Ublox [7]

## 2.6 Sensor Suhu LM35

Sensor suhu LM35 adalah komponen elektronika yang memiliki fungsi untuk mengubah besaran suhu menjadi besaran listrik dalam bentuk tegangan. Sensor Suhu LM35 yang dipakai dalam penelitian ini berupa komponen elektronika yang diproduksi oleh *National Semiconductor*. LM35 memiliki keakuratan tinggi dan kemudahan perancangan jika dibandingkan dengan sensor suhu yang lain, LM35 juga mempunyai keluaran impedansi yang rendah dan linieritas yang tinggi sehingga dapat dengan mudah dihubungkan dengan rangkaian kendali khusus serta tidak memerlukan penyetelan lanjutan.

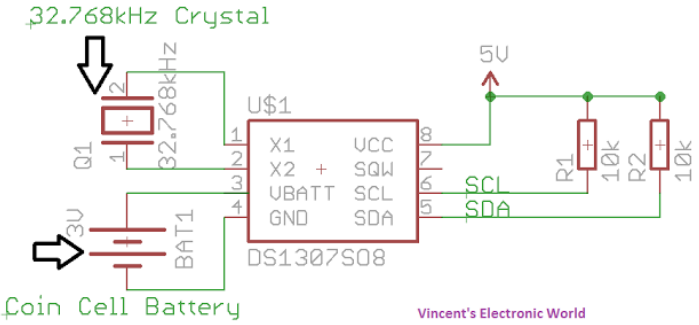



**Gambar 2.10** Sensor Suhu LM35

Meskipun tegangan sensor ini dapat mencapai 30 volt akan tetapi yang diberikan kesensor adalah sebesar 5 volt, sehingga dapat digunakan dengan catu daya tunggal dengan ketentuan bahwa LM35 hanya membutuhkan arus sebesar 60 µA hal ini berarti LM35 mempunyai kemampuan menghasilkan panas (*self-heating*) dari sensor yang dapat menyebabkan kesalahan pembacaan yang rendah yaitu kurang dari 0,5 ºC pada suhu 25 ºC. [8]

## 2.7 RTC (*Real Time Clock*)

RTC (Real time clock) adalah jam elektronik berupa chip yang dapat menghitung waktu (mulai detik hingga tahun) dengan akurat dan menjaga/menyimpan data waktu tersebut secara real time. Karena jam tersebut bekerja real time, maka setelah proses hitung waktu dilakukan output datanya langsung disimpan atau dikirim ke device lain melalui sistem antarmuka. [9]




**Gambar 2.11** RTC

Chip RTC sering dijumpai pada motherboard PC (biasanya terletak dekat chip BIOS). Semua komputer menggunakan RTC karena berfungsi menyimpan informasi jam terkini dari komputer yang bersangkutan. RTC dilengkapi dengan baterai sebagai pensuplai daya pada chip, sehingga jam akan tetap up-to-date walaupun komputer dimatikan. RTC dinilai cukup akurat sebagai pewaktu (timer) karena menggunakan osilator kristal. Banyak contoh chip RTC yang ada di pasaran (pasar genteng, dll) seperti DS12C887, DS1307, DS1302, DS3234. Ada dua buah jenis IC RTC yaitu:

1. DS1307 menggunakan jalur data parallel yang dapat menyimpan data-data detik, menit, jam, tanggal, bulan, hari dalam seminggu, dan tahun valid hingga 2100, 56byte, *battery backed*, RAM (*Random Acces Memory*) *non-volatile* untuk penyimpanan.
2. DS12C887 menggunakan jalur data seri yang memiliki register yang dapat menyimpan data detik, menit, jam, tanggal, bulan, hari dalam seminggu, dan tahun. RTC ini menggunakan bus yang termultipleks untuk menghemat pin. *Timing* yang digunakan untuk mengakses RTC dapat menggunakan *intel timing* atau *motorola timing*. RTC ini dilengkapi dengan pin IRQ untuk kemudahan dalam proses.

## 2.8 Sensor Ultrasonik

Sensor ultrasonik didunia pasaran elektronik juga disebut dengan *Ping sensor* atau *Paralax sensor*. Sensor ultrasonik adalah sensor yang memancarkan sinyal ultrasonik kemudian menangkap kembali sinyal tersebut. Sensor ultrasonik dapat digunakan untuk menghitung jarak dengan mengukur jeda waktu antara sinyal yang dikirim dan diterima. Pada tugas akhir ini sensor ultrasonik digunakan untuk menghitung jarak robot dengan halangan didepan robot. Sensor ini diproses dengan Arduino Mega dan setelahnya akan dikirim ke minimosd melalui komunikasi serial TTL. [10]

# BAB III
# PERANCANGAN DAN REALISASI ALAT

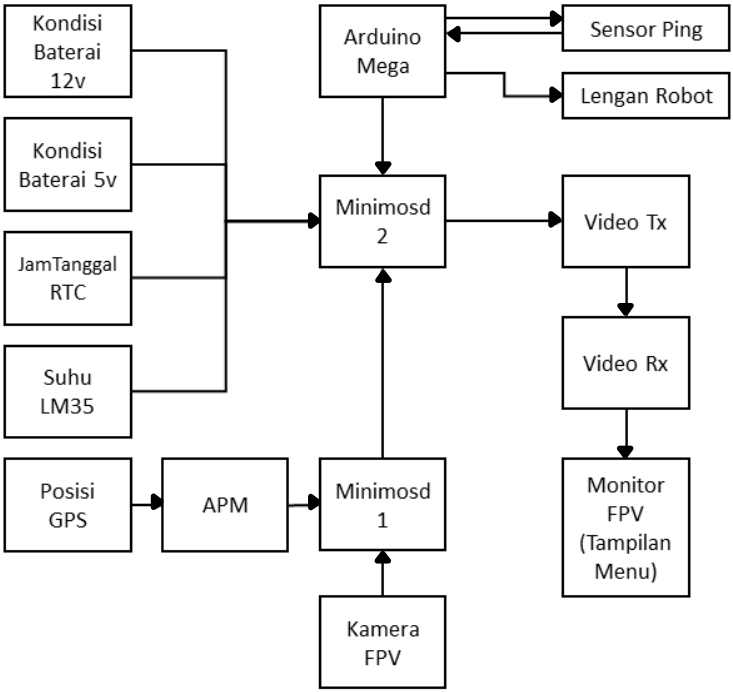
Dalam bab ini akan membahas mengenai perancangan dan realisasi alat yang meliputi diagram blok sistem, perencanaan perangkat keras (*hardware*) dan perangkat lunak (*software*). Hal tersebut guna mewujudkan Tugas Akhir yang berjudul "Pengembangan OSD (*On Screen Display*) dengan Penambahan Menu untuk Aplikasi pada *Semi Autonomous Mobile Robot* dengan Lengan untuk Mengambil Objek". Perancangan alat akan dibahas perbagian disertai dengan gambar *wiring*. Sedangkan penjelasan *software* akan dijelaskan mengenai pembuatan program tampilan pada layar monitor oleh minimosd dengan menggunakan program Arduino, dan perancangan program menggunakan *software* minimosd extra.

Untuk memudahkan dalam pembahasan bab ini akan dibagi menjadi tiga yaitu:

1. Diagram Blok Sistem yang merupakan alur dari cara kerja sistem secara keseluruhan.
2. Perancangan *hardware* (perangkat keras) dengan melakukan pengkabelan minimosd1, penambahan *extra wires* pada minimosd 2, konfigurasi beberapa data sensor pada minimosd, konfigurasi minimosd dengan Arduino Mega yang terdapat pada mobile robot, konfigurasi minimosd dengan APM, konfigurasi sistem FPV.
3. Perancangan *software* (perangkat lunak) meliputi flowchart program arduino, perancangan penampilan karakter, konversi data-data sensor kedalam bentuk karakter untuk ditampilkan pada layar monitor FPV, pembuatan animasi sensor ping dan posisi lengan robot, serta perancangan tampilan awal dan tampilan menu.
4. Realisasi Alat yang merupakan hasil jadi berupa produk dari hasil rancangan *software* dan *hardware*.

## 3.1 Diagram Blok Sistem

Pada tahapan ini dilakukan sebuah penggambaran dari cara kerja keseluruhan sistem dengan menggunakan sebuah diagram blok. Berikut pada gambar 3.1 merupakan garis besar dari diagram blok sistem.




**Gambar 3.1** Blok Diagram Keseluruhan Sistem

Sesuai dengan alur diagram blok pada gambar 3.1, cara kerja dari sistem ini menggunakan beberapa mikrokontroler untuk mendukung beberapa fitur menu sensor yang akan ditampilkan pada layar monitor FPV. Pada gambar terlihat terdapat dua minimosd, minimosd pertama digunakan untuk mengambil data GPS yang dikirim oleh APM yang diprogram menggunakan software gui pemrograman osd yakni minimosd extra. Untuk minimosd yang kedua dilakukan penambahan pin atau *extra wires* untuk mengambil data sensor analog dan level *battery* yang memanfaatkan kaki pin ADC (*Analog to Digital Converter*). Dan untuk pengambilan data waktu dan tanggal menggunakan RTC dengan memanfaatkan kaki pin SDA (*Data* Line) dan SCL (*Clock Line*) pada minimosd tersebut. Sedangkan untuk data sensor ultrasonik dan posisi lengan robot didapat dari Arduino Mega (otak dari *mobile robot*) yang dikirim menggunakan komunikasi serial

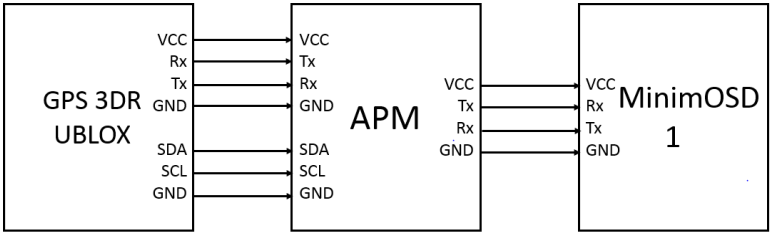
TTL. Data-data yang ditampung oleh minimosd-2 akan diolah kedalam bentuk karakter-karakter yang ditampilkan pada layar monitor FPV. Data sensor temperatur/suhu, data waktu dan tanggal, data level *battery*, data GPS ditampilkan berupa display pada layar, sedangkan data sensor ultrasonik dan posisi lengan robot dianimasikan dalam bentuk level skala perbandingan pada layar. Saat mode biasa dan mode *tracking* nantinya juga akan ditampilkan pada layar monitor. Sistem pengiriman video yang digunakan pada sistem ini yaitu menggunakan *video transmitter*, dan ditangkap oleh *video receiver* yang setelahnya ditampilkan pada layar monitor FPV. Dengan melihat menu data sensor pada monitor kita dapat mengetahui secara *live streaming* keadaan *robot* yang sedang melakukan kerjanya tersebut.

## 3.2 Perancangan *Hardware*

Perancangan *hardware* pengembangan osd ini dengan melakukan penambahan pengkabelan pada minimosd, serta melakukan penggabungan fungsi beberapa komponen seperti sensor suhu, RTC, baterai, GPS, konfigurasi minimosd dengan Arduino Mega, konfigurasi dengan APM serta kamera FPV.

### 3.2.1 Perancangan Pengkabelan Minimosd 1

Pada minimosd yang pertama dilakukan perancangan pengkabelan dengan mengkonfigurasi modul APM yang mengambil data dari GPS 3DR Ublox, setelah itu dikirm ke minimosd 1 untuk diproses kedalam bentuk data visual. Pada minimosd 1 ini diprogram dengan memanfaatkan *platform* yang sudah tersedia di *google* yakni menggunakan software minimosd extra. Berikut pada gambar 3.2 merupakan rancangan pengkabelan dari minimosd pertama.

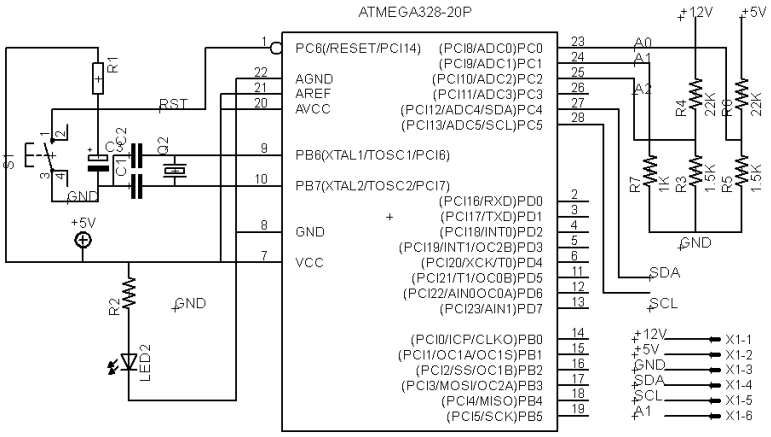



**Gambar 3.2** *Wiring* Minimosd 1

Data GPS yang telah dikonversi menjadi data visual pada minimosd yang pertama ini nantinya akan dikombinasikan dengan data-data sensor yang telah diolah oleh minimosd yang kedua. Kaki pin Video Input pada minimosd pertama dihubungkan dengan kaki pin kamera FPV, sedangkan kaki pin Video Out minimosd 1 dijadikan sebuah inputan dari kaki pin Video Input minimosd 2, yang nantinya data visual akan bertumpukan dan menjadi sebuah data kesatuan yang ditampilkan pada layar monitor FPV.

### 3.2.2 Perancangan *Extra Wires* Minimosd 2

Pada minimosd hanya terdapat kaki pin Tx dan Rx sebagai penerima data dari luar. Secara bentuk *packaging*nya minimosd tersebut hanya dapat menerima data dari modul APM seperti data GPS, data gyro, accelero, dan beberapa jenis kecepatan. Untuk dapat membaca data ADC atau pembacaan sistem I2C (*Inter Integrated Circuit*) maka pada minimosd dilakukan penambahan *extra wires* dengan mensolder kaki pin ATMega 328 SMD dan dijumper pada *board* PCB (*Printed Circuit Board*). Berikut pada gambar 3.3 merupakan perancangan penambahan pengkabelan minimosd tersebut.




**Gambar 3.3** *Extra Wiring* Minimosd 2

Terlihat pada gambar 3.3 bahwa dilakukan penambahan pengkabelan pada minimosd. Hal ini dilakukan agar dapat menambah

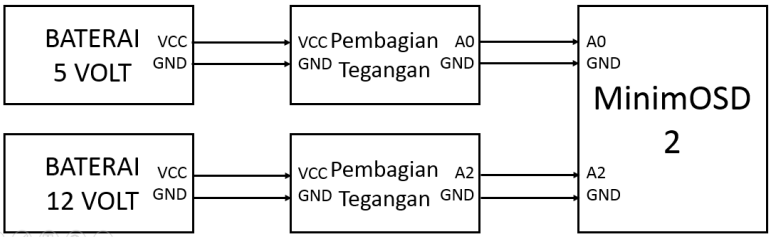
data-data sensor yang berupa ADC dan juga I2C, oleh karena terbatasnya pin pada minimosd yakni hanya Tx dan Rx saja. Dan berikut terlihat pada tabel 3.1 keterangan pin dari gambar tersebut.

**Tabel 3.1** Konfigurasi Pin Minimosd 1

| Pin ATMega 328 | Pin Arduino | Fungsi | Ket. |
|---|---|---|---|
| PORTC.0 | A0 | ADC | Baterai 12v |
| PORTC.1 | A1 | ADC | LM35 |
| PORTC.2 | A2 | ADC | Baterai 5v |
| PORTC.4 | A4 | SDA | RTC |
| PORTC.5 | A5 | SCL | |

### 3.2.3 Perancangan Baterai 5 Volt dan 12 Volt

Untuk mengetahui dan memonitoring tegangan baterai yang bekerja pada sistem *mobile robot* agar tidak terjadi kehabisan baterai saat proses kerja, maka dilakukan pengambilan data baterai sistem dengan menggunakan pembacaan ADC (*Analog to Digital Converter*). Pengkabelan baterai ini dengan menggunakan *extra wires* yang terhubung dengan pin A0 dan A2 seperti yang telah diulas sebelumnya pada gambar 3.3. Berikut pada gambar 3.4 ilustrasi dari pengkabelan batterai pada minimosd.




**Gambar 3.4** *Wiring* Baterai 5 volt dan 12 volt

Terlihat pada gambar 3.4 desain pengkabelan baterai 12 volt dan 5 volt dengan menggunakan *extra wires* yang dirangkai bersama resistor 22kohm dan 1,5kohm. Kaki Vcc kedua baterai dihubungkan dengan 22kohm agar terjadi pembagian tegangan yang sangat kecil dan meminimalisir arus yang terlalu besar. Hal ini agar rangkaian tersebut tidak mengalami kerusakan oleh adanya arus bocor yang disebabkan

oleh baterai. Berikut merupakan perhitungan dari tegangan baterai yang masuk pada kaki pin ATMega 328.

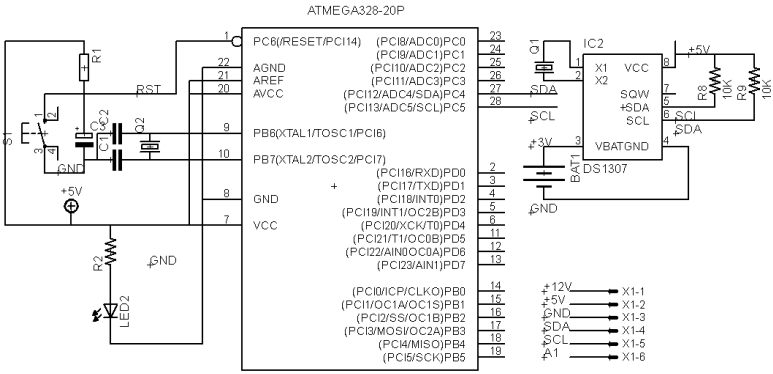
- Diketahui:
  R1 = 22K
  R2 = 1,5K
  V12 = 12 volt
  V5 = 5 volt
- Rumus:

$$V_{A0} = \frac{R_2}{R_1+R_2} V_{12} \text{.....(3.1)}$$

$$V_{A2} = \frac{R_2}{R_1+R_2} V_5 \text{.....(3.2)}$$

### 3.2.4 Perancangan Rangkaian RTC

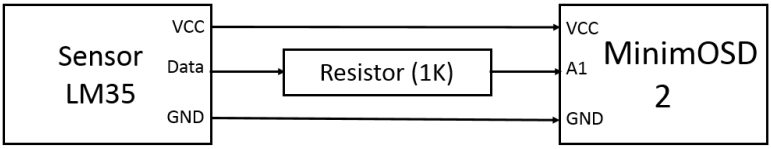
Pada tahapan ini dilakukan sebuah perancangan rangkaian RTC (*Real Time Clock*) guna sebagai pelengkap sistem tersebut dengan mengetahui jam dan waktu robot melakukan proses kerjanya. Untuk rangkaian RTC ini menggunakan modul RTC Arduino DS1307 24C32. Modul RTC ini menggunakan pengiriman I2C yaitu Data Line dan Data Clock atau SDA SCL. Oleh karena pada minimosd tidak ada tempat untuk pin SDA dan SCL, maka sama halnya untuk rancangan baterai, disini dilakukan penambahan pengkabelan pada pin PORTC.4 dan PORTC.5 sebagai masukan data dan clock pada ATMega 328. Berikut pada gambar 3.5 merupakan *wiring* diagram dari rangkaian tersebut.




**Gambar 3.5** *Wiring* RTC

### 3.2.5 Perancangan Rangkaian Sensor Suhu LM35

Penambahan pengkabelan pada minimosd juga dilakukan untuk sensor suhu LM35 ini. Sensor suhu atau temperatur yang dipasang pada *mobile robot* ini berguna untuk mengetahui kondisi sekitar *mobile* robot dalam keadaan panas atau dingin. Dengan mengetahui suhu pada *mobile robot* maka dapat digunakan sebagai indikator performa robot tersebut mengalami gangguan atau tidak. Berikut pada gambar 3.6 merupakan wiring dari rangkaian sensor suhu LM35 dengan minimosd.



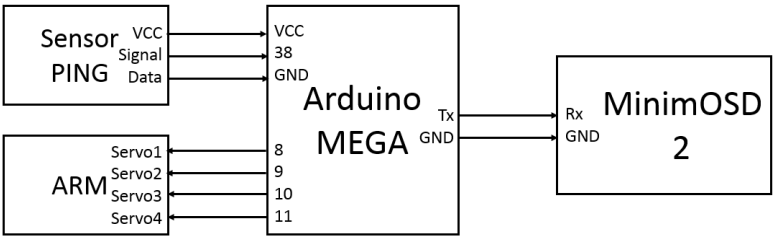

**Gambar 3.6** *Wiring* LM35

Terlihat pada gambar 3.6 tegangan output dari sensor suhu LM35 masuk ke pin ADC A1, tetapi sebelum memasuki kaki pin A1 pada output sensor dihubungkan dengan resistor 1Kohm. Resistor disini hanya berfungsi untuk memperkecil nilai arus yang masuk ke ATMega 328. Sensor suhu LM35 ini memiliki parameter bahwa setiap kenaikan 1ºC tegangan keluarannya naik sebesar 10mV dengan batas maksimal keluaran sensor adalah 1,5V pada suhu 150°C. Berikut merupakan rumus dari tegangan output LM35.

$V_{out} = 10mV/°C \; x \; T_{ruang}$ ……..(3.3)

### 3.2.6 Perancangan Arduino Mega dengan Minimosd

Arduino Mega yang merupakan otak dari *mobile robot* disini digunakan untuk mengambil data sensor ultrasonik dan memberikan instruksi pada lengan robot. Data sensor ultrasonik yang didapat oleh Arduino Mega berupa jarak robot dengan sebuah objek nantinya akan dikirim ke minimosd 2 menggunakan pengiriman komunikasi serial TTL. Begitu pula perintah untuk menggerakkan lengan robot berupa sudut, pun dikirim bersama data sensor ultrasonik menuju minimosd 2. Pada minimosd 2 nantinya data yang diperoleh dari Arduino Mega ini akan diolah menjadi sebuah animasi berupa level skala perbandingan dengan data sebenarnya. Dan berikut pada gambar 3.7 merupakan rancangan pengkabelan antara Arduino Mega, Sensor Ultrasonik, posisi lengan robot, serta minimosd.



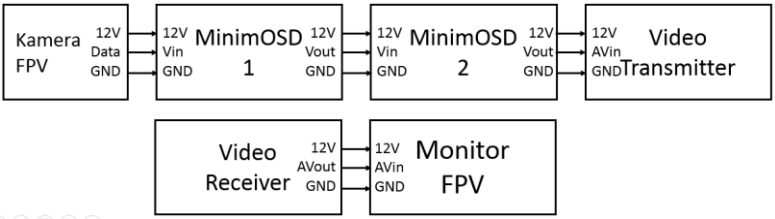

**Gambar 3.7** *Wiring* Arduino Mega dan Minimosd

Pada tugas akhir ini, minimosd 2 hanya menerima data dari Arduino Mega yang ada pada *mobile robot* dan setelah itu mengolah data tersebut kedalam bentuk karakter untuk ditampilkan pada layar monitor FPV.

### 3.2.7 Perancangan Kombinasi Minimosd dan Sistem FPV

Pada tahap ini dilakukan sebuah perancangan kombinasi 2 (dua) minimosd dan penggabungan data visual kamera AV dengan data visual minimosd, yang setelahnya dikirim menggunakan *video transmitter* menuju *video receiver* untuk ditampilkan pada layar monitor FPV. Kaki pin video pada kamera AV dihubungkan ke kaki pin Vin (Video input) pada minimosd 1, sedangkan untuk kaki pin Vout (Video Out) pada minimosd 1 dihubungkan ke kaki pin Vin pada minimosd 2. Setelah itu kaki pin Vout pada minimosd 2 merupakan kaki pin input yang akan dihubungkan ke *video transmitter*. Data yang dikirm ini adalah data kombinasi dari kamera AV, minimosd 1, dan minimosd 2. Data yang ditangkap oleh video receiver berupa data gabungan yang sudah diisi oleh beberapa menu sensor dan akan ditampilkan pada layar monitor FPV guna sebagai monitoring *live streaming*. Terdapat perbedaan antara minimosd 1 dan minimosd 2 yakni pada minimosd 1 *software complier* yang digunakan menggunakan *compiler* yang telah disediakan oleh *google* berupa tampilan gui, sehingga kita tidak dapat melakukan pengembangan isi program, selain itu pada minimosd 1 ini hanya menggunakan kaki pin Tx Rx untuk menerima data, minimosd ini hanya bias terkoneksi dengan ArduPilot Mega yang menampung beberapa data sensor seperti GPS dan dikirim dengan pengiriman serial. Sedangkan untuk minimosd 2 dilakukan sebuah pengembangan agar terdapat beberapa fitur menu sensor tambahan yang menggunakan ADC ataupun I2C. Pengembangan yang dilakukan yakni dengan melakukan

penambahan pin extra pada minimosd 2 yang sudah dijelaskan pada subbab sebelumnya. Berikut pada gambar 3.8 merupakan rancangan pengkabelan dengan melakukan kombinasi antara dua minimosd dan pengiriman dengan sistem FPV.




**Gambar 3.8** *Wiring* Kombinasi Minimosd dan Sistem FPV

Terlihat pada gambar 3.8 merupakan kombinasi dari dua minimosd dan pengiriman data visual menggunakan *video transmitter* dan *receiver* dengan sistem FPV.

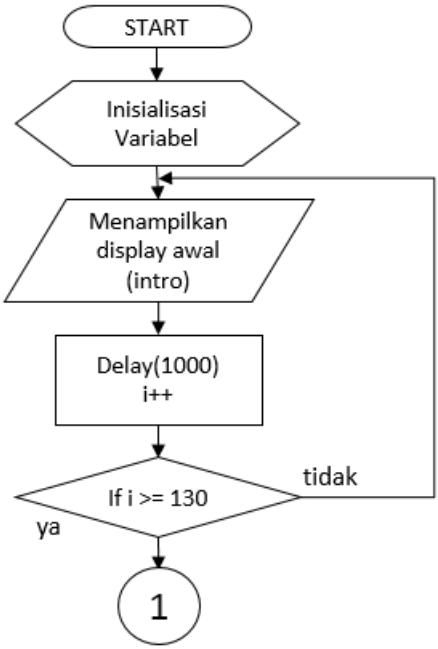
## 3.3 Perancangan *Software*

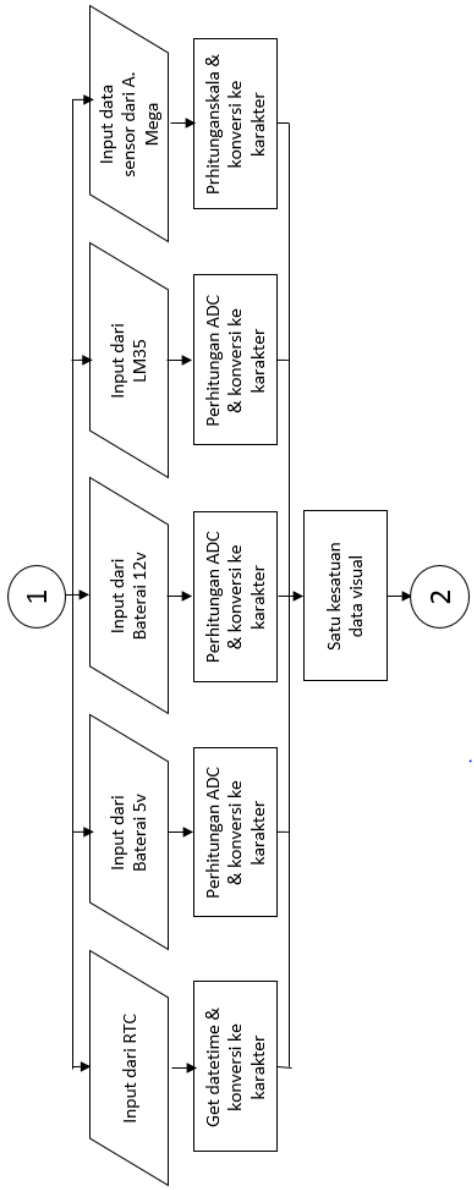
Pada tahap perancangan software atau perangkat lunak pada tugas akhir ini terdiri dari 2 bagian yakni perancangan perangkat lunak pada minimosd menggunakan compiler Arduino dan perancangan perangkat lunak menggunakan *compiler* dari *google* berupa tampilan GUI yaitu minimosd extra. Untuk perancangan perangkat lunak menggunakan Arduino terdiri dari beberapa bagian program dengan penjelasan setiap bagian program. Sedangkan perancangan perangkat lunak yang menggunakan minimosd extra hanya dengan menyusun data yang akan ditampilkan pada monitor, karena data-data sensor telah disediakan dan sangat mudah untuk penyusunan data sensor pada grafik layar monitor.

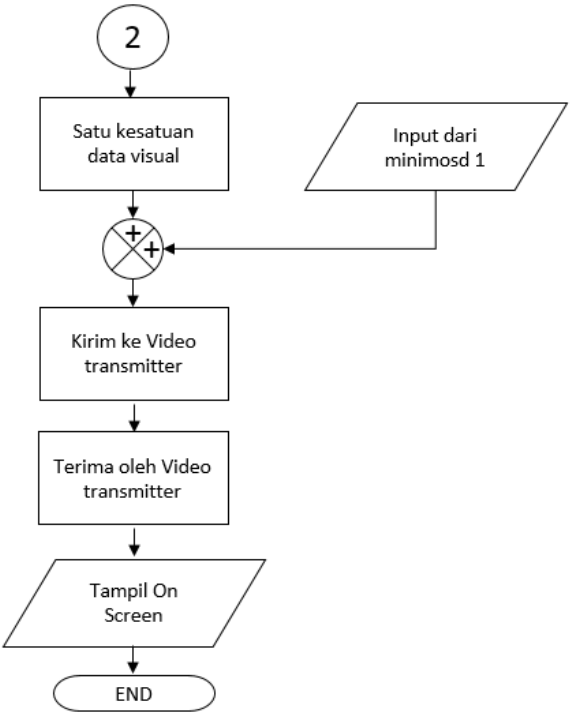
Untuk mengetahui cara kerja program yang digunakan pada sistem, maka berikut akan dijelaskan mengenai flowchart dari program Arduino yang telah dibuat dengan penambahan beberapa fitur menu. Sedangkan untuk pemrograman dengan minimosd extra tidak dijelaskan flowchart kerjanya, karena bukan merupakan fokusan dari tugas akhir yang dibuat dan program yang terlalu sederhana dengan tampilan GUI.

### 3.3.1 Flowchart Program Arduino

Pada tahapan ini menjelaskan bagaimana alur kerja program Arduino berjalan yang digambarkan dengan flowchart. Terlihat pada gambar 3.10 program diawali dengan inisialisasi atau deklarasi variabel, tujuannya untuk mempermudah dalam proses kerja program tersebut. Setelah melakukan inisialisasi dan deklarasi variabel barulah program melakukan proses pertamanya untuk menampilkan display awal atau intro yang dikirim lewat Vtx menuju Vrx layar monitor FPV. Dengan selang waktu beberapa detik setelah itu program tersebut bertugas untuk menangkap dan memproses semua data yang diterima. Disini dilakukan proses konversi menjadi sebuah kumpulan karakter yang nantinya membentuk satu kesatuan dengan beberapa menu sensor. Data yang telah diproses oleh minimosd 2 akan digabung dengan data dari minimosd 1 yang telah mengambil data dari APM dan juga digabung dengan data visual kamera. Penggabungan data ini yang akan dikirm dan ditampilkan pada layar monitor FPV.




1
Input dari RTC
Get datetime & konversi ke karakter
Input dari Baterai 5v
Perhitungan ADC & konversi ke karakter
Input dari Baterai 12v
Perhitungan ADC & konversi ke karakter
Input dari LM35
Perhitungan ADC & konversi ke karakter
Input data sensor dari A. Mega
Prhitunganskala & konversi ke karakter
Satu kesatuan data visual
2




**Gambar 3.9** Flowchart Program Arduino

### 3.3.2 Perancangan Penampilan Karakter

Perancangan untuk proses pengkonversian data-data sensor menjadi sebuah karakter ini menggunakan IC Max7456. Didalam IC ini terdapat 256 data karakter yang semuanya sudah tersedia dari pabrikannya. Dengan melakukan proses konversi data sensor yang berupa integer atau float menjadi sebuah array char (karakter yang berupa data array). Begitu pula dengan data sensor yang didapat dari Arduino Mega melalui pengiriman serial, baik yang berupa string atau char, semuanya akan dikonversi menjadi sebuah array sehingga 1 data karakter sama dengan 1 byte atau 8bit data. Misalkan terdapat data string a = “aku”, maka data ini akan dipecah untuk diambil satu persatu bagian dan dimasukkan kedalam 1 char array, sehingga data menjadi char b[4] = “aku” yang mengartikan char array tersebut mempunyai 4

data dengan alamat dari 0-3. Berikut pada gambar 3.10 merupakan bagian dari program Arduino untuk menampilkan karakter tersebut.

```
void MAX7456_WriteString_P(const char *string, int Adresse)
{
        uint8_t xx = 0;
        char c;
        while((c = (char)pgm_read_byte(&string[xx++])) != 0)
        {
                screen[Adresse++] = c;
        }
}
```

**Gambar 3.10** Program Menulis String

Untuk sintaks penulisan maka ditulis dengan "MAX7456_WriteString_P (data,134);", dengan sintaks berikut maka variable data yang berupa array char akan dilakukan pembacaan atau pengambilan data byte satu persatu dan setelah itu melakukan penulisan pada koordinat posisi ke 13. Penulisan akan dilakukan sampai tidak adanya data pada variable tersebut. Misalkan variabel data = "aku" dengan posisi = 134, maka output yang akan ditampilkan pada layar akan seperti pada gambar 3.11 dibawah ini.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | A | K | U | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Gambar 3.11** Koordinat *Screen*

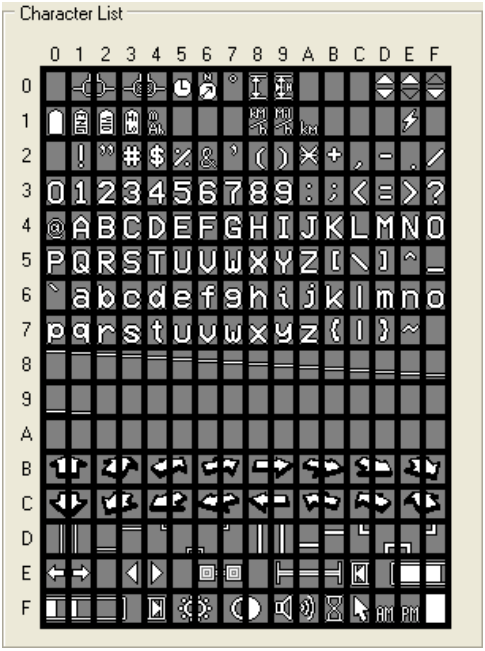
Jika kita ingin menuliskan sebuah huruf atau kata pada posisi kooordinat x dan y tertentu maka perhitungannya seperti dibawah ini.

- Diketahui:
  X = 17  Y = 5
  Xmax = 30  Ymax = 16

- Rumus:
  $Posisi = (Y - 1)X_{max} + X$……..(3.4)
  = (5-1)30 + 17
  = 4*30 + 17
  = 137

Maka sintaks penulisan untuk menampilkan karakter yakni "MAX7456_WriteString_P (data,137);".

Sedangkan karakter yang dapat ditampilkan pada layar monitor hanya berisi 256 data karakter, oleh karena keterbatasan memori pada IC Max7456 yang hanya bisa menampung 256 data. Berikut pada gambar 3.12 kumpulan data karakter tersebut.




**Gambar 3.12** Simbol minimosd

### 3.3.3 Perancangan Program ADC Baterai

Pada perancangan program untuk baterai 5 volt dan 12 volt ini menggunakan pembacaan ADC. Data analog yang dihasilkan oleh baterai masuk ke kaki pin ADC minimosd (ATMega 328), data yang terbaca di kontroler ini berupa data digital, sehingga data digital tersebut akan dikalkulasi untuk dicari atau dikembalikan ke data analog sebenarnya. Data analog yang telah dikalkulasi nantinya akan dikonversi lagi menjadi sebuah char array agar dapat diolah oleh Max7456 dan ditampilkan di layar monitor FPV. Berikut merupakan potongan program pembacaan dan kalkulasi ADC.

```
VDig5 = analogRead(VbatOSDPin); //Baca nilai digital V5
VAna5 = (float)VDig5* 0.0783; //kalkulasi V5 analog
VDig12 = analogRead(VbatVideoPin); //Baca nilai digital V12
VAnal2 = (float)VDig12 * 0.0783; //kalkulasi V12 analog
```

Program diatas didapat dari kombinasi perhitungan pembagian tegangan dan pembacaan ADC. Berikut rumus dasar dari perhitungan tersebut.

- Diketahui:
  R1 = 22K  R2 = 1,5K
  V12 = 12 volt  V5 = 5 volt
  Nmax = 1023  Vref = 5.11 volt

- Rumus:

$$V_{A0} = \frac{R_2}{R_1 + R_2} V_{12} = \frac{V_{dig12}}{N_{max}} V_{ref}$$

$$V_{12} = \frac{(R_1+R_2)V_{dig12}V_{ref}}{R_2 N_{max}} \text{.....(3.5)}$$

$$= \frac{(1{,}5 + 22)V_{dig12}.5{,}11}{1{,}5.1023}$$

$$= 0{,}0783V_{dig12}$$

$$V_{A2} = \frac{R_2}{R_1 + R_2} V_5 = \frac{V_{dig5}}{N_{max}} V_{ref}$$

$$V_5 = \frac{(R_1+R_2)V_{dig5}V_{ref}}{R_2 N_{max}} \text{.....(3.6)}$$

$$= \frac{(1{,}5 + 22)V_{dig5}.5{,}11}{1{,}5.1023}$$
$$= 0{,}0783V_{dig5}$$

Setelah melakukan pembacaan nilai ADC dan melakukan kalkulasi sehingga hasil yang diperoleh adalah data analog yang dihasilkan oleh baterai, maka setelah itu dilakukan pengkonversian data *float* tersebut menjadi sebuah array char untuk ditampilkan pada *screen*. Sintaks program seperti dibawah ini.

```
dtostrf(VAna5,2,2,disp); //konversi double to char (2 angka belakang koma)
MAX7456_WriteString(disp,413); //tulis data ke screen
dtostrf(VAnal2,2,2,disp); //konversi double to char
MAX7456_WriteString(disp,443); //tulis data ke screen
```

Data baterai berupa float yang sudah dikalkulasi akan dilakukan sebuah perbandingan dengan konstanta tertentu dan dengan permisalan yang mempunyai level tingkatan untuk menggambarkan keadaan isi baterai dalam kondisi penuh atau tidak. Dan program permisalan tersebut seperti dibawah ini.

```
     if (VAnal2 < 9.5) screenBuffer[0] = 0xb4;//icon baterai kosong
else if (VAnal2 < 10)  screenBuffer[0] = 0xb5;//i.baterai hampir habis
else if (VAnal2 < 10.5)screenBuffer[0] = 0xb6;//i.baterai 25%
else if (VAnal2 < 11)  screenBuffer[0] = 0xb7;//i.baterai 50%
else if (VAnal2 < 11.5)screenBuffer[0] = 0xb8;//i.baterai 75%
else if (VAnal2 < 12)  screenBuffer[0] = 0xb9;//i.baterai 100%
else                   screenBuffer[0] = 0xba; //i.baterai full
MAX7456_WriteString(screenBuffer,442);//tulis data ke screen
```

Dan berikut pada tabel 3.2 merupakan penjelasan dari simbol baterai dengan level tegangan baterai.

**Tabel 3.2** *Display* Level Baterai

| Simbol | Level (volt) | Keterangan |
|---|---|---|
|  | < 9.5 | Baterai Kosong |
|  | 9.5 ≤ V < 10 | Hampir Habis |

| | | |
|---|---|---|
|  | $10 \leq V < 10.5$ | Baterai 25% |
|  | $10.5 \leq V < 11$ | Baterai 50% |
|  | $11 \leq V < 11.5$ | Baterai 75% |
|  | $11.5 \leq V < 12$ | Baterai 100% |
|  | $V > 12$ | Full |

### 3.3.4 Perancangan Program RTC

Untuk perancangan program RTC (Real Time Clock) menggunakan metode pengiriman I2C (*Inter Integrated Circuit*). Pemrograman RTC yang menggunakan IC DS1307 dengan pengiriman I2C ini menggunakan *library* yang sudah tersedia dan dapat didownload di web resmi Arduino. Dan berikut merupakan potongan program untuk pengambilan data waktu dan tanggal.

```
#include <RTClib.h>
#include <Wire.h>
RTC_DS1307 rtc;
DateTime now1;
now1 = rtc.now();
String a = String(now1.day());
a.toCharArray(h,3);
String b = String(now1.month());
b.toCharArray(bb,3);
String c = String(now1.year());
c.toCharArray(t,5);
String d = String(now1.hour());
d.toCharArray(j,3);
String e = String(now1.minute());
e.toCharArray(m,3);
String f = String(now1.second());
f.toCharArray(dd,3);
```

Pada potongan program diatas terlihat bahwa untuk pemrograman RTC menggunakan *library* “RTClib” dan header “Wire”

untuk mengaktifkan kaki pin I2C. Untuk pengambilan data jam dan tanggal dilakukan terus menerus, maksudnya inisialisasi DateTime seperti pada program diatas ditaruh pada program perulangan atau "loop", agar *clock* dapat terus bekerja dan data akan dikirim setiap *clock*-nya. *Clock* atau siklus kerja tersebut dipengaruhi oleh *crystal* dari Atmega dan RTC itu sendiri. Pada program tersebut data jam dan tanggal dimasukkan satu persatu kedalam sebuah variabel dan setelah itu dikonversi kedalam bentuk char array agar dapat ditampilkan pada layar monitor.

```
if (a.toInt()<10){
  MAX7456_WriteString(nol,19);//"0"
  MAX7456_WriteString(h,20);
}
else {
  MAX7456_WriteString(h,19);
}
```

Sedangkan untuk potongan program diatas merupakan permisalan yang digunakan untuk menambahkan angka 0 didepan angka yang diambil jika angka tersebut kurang dari 10. Oleh karena jika angka yang diambil kurang dari 10 maka yang tertulis pada layar nantinya hanya satu angka itu saja, tanpa diawali dengan angka 0. Penambahan angka 0 ini digunakan hanya untuk mengikuti standart penulisan jam dan tanggal saja.

### 3.3.5 Perancangan Program LM35

Pada tahapan perancangan program sensor suhu LM35 ini sama halnya dengan perancangan program baterai yakni menggunakan sistem pembacaan ADC 10bit yang sudah tersedia pada chip ATMega 328. Sensor suhu yang akan ditampilkan pada layar monitor ini sebelumnya dilakukan sebuah pengkalkulasian seperti pada potongan program dibawah ini.

```
VDsuhu = analogRead(suhuPin); //baca nilai digital Vout suhu
derSuhu = (float)VDsuhu * 0.4888; //kalkulasi nilai suhu
```

Perhitungan dari program diatas didapat dari hasil kalkulasi antara tegangan output sensor setiap kenaikan 1°C dengan nilai bit data mikro. Berikut merupakan perumusannya.

- Diketahui:
  Nmax = 1023
  Vref = 5.11 volt

- Rumus:

$$V_{out} = \frac{10mV}{°C} x\, T_{ruang} = \frac{V_{dig}}{N_{max}} V_{ref}$$

$$T_{ruang} = \frac{100.V_{dig}}{N_{max}} V_{ref} \text{.....(3.7)}$$

$$= \frac{100.V_{dig}}{1023} 5$$

$$= 0{,}4888 V_{dig}$$

Sedangkan untuk menampilkan data sensor suhu yang telah dikalkulasi tersebut, sebelumnya harus dilakukan pengkonversian data float tersebut menjadi bentuk data char array seperti pada potongan program dibawah ini.

```
dtostrf(derSuhu,2,0,disp); //convert float to char array
MAX7456_WriteString(disp,386); //tulis data ke screen
screen[388] = 0xb0; //tulis simbol drajat
```

Program diatas juga digunakan untuk menampilkan bentuk simbol derajat pada screen yakni dengan pemanggilan karakter '°' pada alamat ke 0xb0 atau sama dengan alamat ke 176 dengan posisi ke 388 pada screen.

### 3.3.6 Perancangan Program Sensor Ultrasonik dan Posisi Lengan Robot dengan Pengiriman Serial

Pada tahap ini sensor ultrasonik dan posisi lengan robot diolah oleh Arduino Mega dan dikirim ke minimosd melalui pengiriman serial untuk diolah dan dikonversi menjadi sebuah kesatuan karakter membentuk skala perbandingan dengan tingkatan level tertentu. Program pada minimosd menggunakan metode *interrupt receiver* atau bisa dikatakan sebagai *serial event*, metode ini digunakan karena pengerjaan instruksi yang dilakukan oleh Arduino atau mikro yang sekuensial atau runtut yakni mengeksekusi berdasarkan urutan penulisan program. Dengan menggunakan serial event ini, maka saat pengerjaan atau pembacaan data-data sensor dan penulisan karakter tidak perlu menunggu adanya data serial yang masuk. Data serial bias masuk kapan

saja tanpa menggangu sederatan instruksi yang lain. Berikut merupakan potongan dari pemrogram serial event di Arduino.

```
void serialEvent() { //fungsi interrupt receiver
  while (Serial.available()) { //saat serial aktif
    if(Serial.available()>0) { //jika ada data masuk
      char inChar = (char)Serial.read(); //masukkan data ke var inchar
      inputString += inChar; //mengumpulkan data per-byte
      if (inChar == '#') { //jika data berup '#'
        stringComplete = true; //boolean}}}}
```

Program diatas digunakan untuk menampung data yang diterima oleh kaki pin Rx. Setiap *byte* data yang terbaca akan dimasukkan kedalam suatu variabel dan ketika data yang terbaca berupa simbol '#', maka variabel tersebut akan berhenti menampung data atau data ditampung lagi pada keadaan awal. Dari variabel yang sudah berisi kumpulan data tersebut, maka akan dilakukan pemisahan data per bagian atau bisa dikatakan proses *parsing* data untuk memperoleh data sensor ultrasonik dan data posisi lengan robot. Contoh dari program *parsing* data dapat dilihat seperti berikut ini.

```
for(i=1;i<inputString.length();i++){ //perulangan sepanjang var yang didapat
  if ((inputString[i] == '*') || (inputString[i] == ';') || (inputString[i] == '#'))
  { //ketika data ke-x = '*' atau ';' atau '#'
    j++; //menambahkan data ke-
    datac[j]=""; //string kosong
  }
  else //selain diatas
  {
    datac[j] = datac[j] + inputString[i]; //mengambil bagian data ke-
  }}}
```

Setelah memilah atau memarsing data yang didapatkan dari *interrupt* serial tersebut yang berupa data sensor ultrasonik dan data posisi lengan robot, maka akan dilakukan proses penggambaran animasi berupa skala perbandingan dengan tingkatan level tertentu. Berikut potongan program pengolah data tersebut.

```
for(i=0;i<3;i++){ //perulangan sbanyak x=3
   if (tetal>=0 && tetal<90){//jika sudut antara 0-90
     al = tetal/9;//sudut dibagi dengan 10 gambar
     screen[196+i]=5+al;//tulis data ke screen
   }
```

Terlihat pada potongan program diatas bahwa sudut yang diperoleh dibagi dengan 9. Angka 9 tersebut menunjukkan adanya 10

bagian simbol yang akan digambarkan. Berikut pada tabel 3.3 merupakan gambar dan penjelasan dari 10 bagian gambar tersebut.

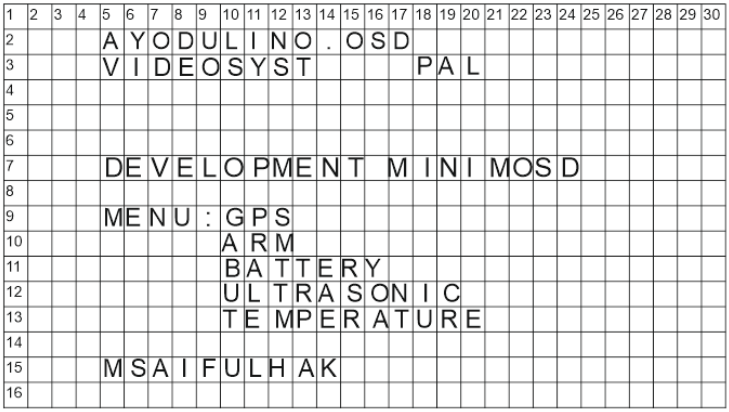
**Tabel 3.3** Display Posisi Lengan Robot

| Simbol | Sudut (°) |
|---|---|
|  | 0 - 9 |
|  | 10 - 18 |
|  | 19 - 27 |
|  | 28 - 36 |
|  | 37 - 45 |
|  | 46 - 54 |
|  | 55 - 63 |
|  | 64 - 72 |
|  | 72 - 81 |
|  | 82 - 90 |

Untuk skala perbandingan data sensor ultrasonik sendiri yakni 1:30 artinya 1 simbol pada layar mewakili jarak 30 cm pada keadaan sebenarnya. Seperti kalkulasi data posisi lengan robot, untuk kalkualsi data sensor ultrasonik pun demikian, data sensor yang didapat dari *receiver* dibagi dengan 30 dan setelah itu akan digambarkan animasi jarak pada layar monitor.

### 3.3.7 Perancangan Tampilan Awal

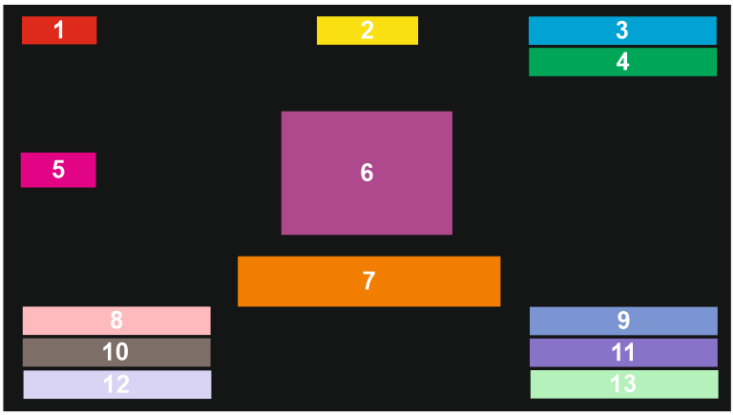
Pada tahap ini dilakukan perancangan tampilan awal atau *display intro* pada layar monitor. Berikut pada gambar 3.13 perancangan tampilan awal yang didesain menggunakan *software corel draw*.




**Gambar 3.13** *Display Intro*

### 3.3.8 Perancangan Tampilan Menu Sensor

Perancangan tampilan menu untuk menata posisi fitur menu yang berupa data sensor. Data-data sensor hasil kalkulasi tersebut ditampilkan semuanya pada satu layar dengan posisi-posisi tertentu. Disini tampilan menu-menu sensor dijadikan satu kesatuan dengan koordinat posisi yang sudah ditentukan. Berikut pada gambar 3.14 merupakan rancangan dari tata letak sensor.




**Gambar 3.14** *Display Menu*

Keterangan:

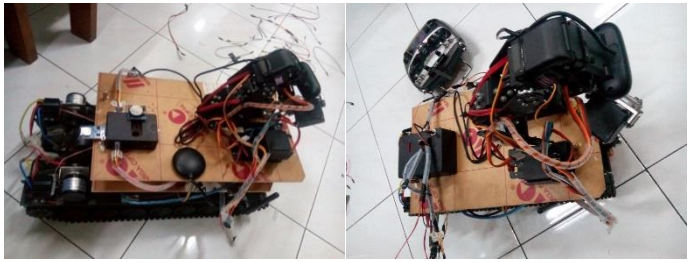
1. Satelit
2. Heading Kompas / Derajat Mata Angin
3. Tanggal, Bulan, dan Tahun
4. Jam, Menit, dan Detik
5. Kecepatan
6. Posisi Lengan Robot
7. Jarak / Ultrasonik
8. Altitude
9. Temperatur
10. Latitude
11. Baterai 5 volt
12. Longitude
13. Baterai 12 volt

## 3.4 Realisasi Alat

Pada tahap ini merupakan tahapan akhir dari bab 3 yakni merealisasikan sebuah sistem menjadi sebuah bentuk produk sesuai dengan hasil perancangan *hardware* dan *software* yang telah dijelaskan sebelumnya. Dan berikut pada gambar 3.16 merupakan bentuk jadi produk yang telah dibuat.

**Gambar 3.15** Produk Minimosd

Setelah alat direalisasikan, barulah mengimplementasikan alat tersebut untuk digunakan pada *Semi Autonomous Mobile Robot* sesuai dengan judul tugas akhir yang telah disusun ini. Dan berikut peletakan alat seperti pada gambar 3.16.



**Gambar 3.16** Implementasi Produk pada *Mobile Robot*

Pada gambar 3.16 terlihat bahwa peletakan produk tepat diatas *mobile robot*. Selain realisasi sistem yang berupa *hardware*, juga dilakukan realisasi sistem dari segi *software*. Realisasi software berupa tampilan *display* awal dan tampilan menu yang sebelumnya telah dilakukan perancangan. Realisasi *software* seperti pada gambar 3.17 dibawah ini.


**Gambar 3.17** Realisasi *Display* Awal

Dan untuk realisasi display menu seperti pada gambar 3.18, terdapat beberapa menu sensor dengan tampilan satu kesatuan antara lain display *arm*, ping, RTC, baterai, temperatur, dan GPS.


**Gambar 3.18** Realisasi *Display* Menu

*--Halaman ini sengaja dikosongkan--*

# BAB IV
# PENGUJIAN DAN ANALISA SISTEM

Bab ini menjelaskan tentang hasil pengujian dan analisa dari perancangan sistem yang telah dilakukan dan dijelaskan pada bab sebelumnya. Pengujian dan analisa yang dilakukan menjadi beberapa bagian yang akan dijelaskan pada sub bab berikut ini.

## 4.1 Pengujian Minimosd

Pengujian dan analisa minimosd dibagi menjadi beberapa tahap pengujian. Berikut akan dijelaskan pengujian dari sistem tersebut.

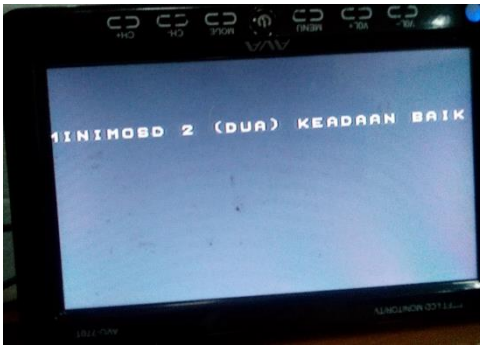
### 4.1.1 Pengujian Minimosd 1

Pada tahap ini dilakukan pengujian minimosd dalam keadaan baik atau tidak. Pengujian dilakukan dengan cara memprogram minimosd menggunakan Arduino uno tanpa chip. Pengujian minimosd ini dilakukan pada minimosd yang pertama dulu, berikut pada gambar 4.1 merupakan hasil dari pengujian alat tersebut.


**Gambar 4.1** Uji Minimosd 1

### 4.1.2 Pengujian Minimosd 2

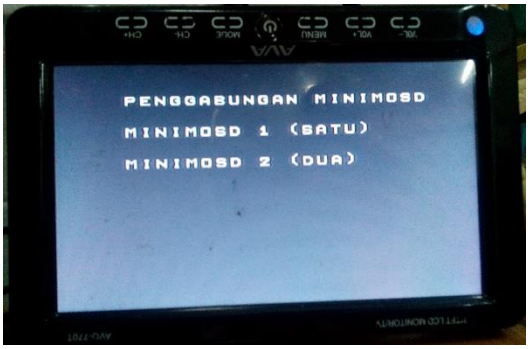
Sedangkan tahap ini merupakan pengujian minimosd yang kedua, sama seperti pengujian yang pertama yakni pengujian dilakukan dengan menggunakan program Arduino uno tanpa chip. Berikut pada gambar 4.2 merupakan hasil dari pengujian alat tersebut.




**Gambar 4.2** Uji Minimosd 2

### 4.1.3 Pengujian Penggabungan Data Minimosd

Setelah melakukan pengujian minimosd satu dan dua, maka selanjutnya dilakukan penggabungan data dua minimosd untuk mengetahui bisa atau tidaknya dilakukan penggabungan dua data minimosd. Dan berikut pada gambar 4.3 merupakan hasil pengujian penggabungan data dua minimosd.




**Gambar 4.3** Kombinasi Dua Minimosd

Tetapi dalam penggabungan data dua minimosd tersebut tidak boleh ada delay waktu dalam pemrogramannya, agar data yang dihasilkan pada saat pertama kali dinyalakan bersamaan dan tidak terdapat layar semut dua kali.

### 4.1.4 Pengujian Data Karakter Max7456

Sesuai dengan yang dijelaskan pada bab sebelumnya, bahwa untuk penulisan karakter pada screen terbatas hanya ada 256 data karakter. 256 data tersebut dipanggil atau ditulis dengan menuliskan alamat dari karakter tersebut. Dan pada gambar 4.4 merupakan 256 data karakter yang dapat dituliskan.


**Gambar 4.4** Data Karakter

Misalkan untuk pemanggilan atau penulisan data karakter "." yang berada pada alamat ke-47 dan akan diletakkan pada koordinat (10,5), berarti perhitungannya:

- Diketahui:
  Xmax = 30 Ymax = 16
  X = 10 Y = 5
  N = 47

- Jawab:
  $n = N - 1 = 47 - 1 = 46$
  $xy = (y - 1)N_{max} + x$
  $= (5 - 1)30 + 10$
  $= 130$
  $s[xy] = n$
  $s[130] = 46$

- Keterangan:
  Xmax = koordinat x maksimal
  Ymax = koordinat y maksimal
  X = posisi x tujuan
  Y = posisi y tujuan
  N = alamat karakter hitungan dari angka-1
  n = alamat karakter hitungan dari 0
  xy = kalkulasi posisi x,y
  s = screen, untuk menampilkan karakter

### 4.1.5 Pengujian Minimosd dengan Kamera

Setelah melakukan pengujian dua minimosd, pengujian kombinasi data, dan pengujian data karakter, barulah dilakukan pengujian penggabungan data dua minimosd dengan kamera FPV. Tujuan pengujian ini agar dapat mengetahui bisa atau tidaknya dilakukan kombinasi data dari minimosd dan kamera. Dan melihat perbedaan antara penggabungan tersebut. Berikut pada gambar 4.5 adalah data minimosd tanpa kamera dan gambar 4.6 adalah hasil dari pengujian menggunakan kamera.


**Gambar 4.5** Data Minimosd Tanpa Kamera


**Gambar 4.6** Kombinasi Minimosd dengan Kamera

Jika dilihat dari gambar 4.5 dan gambar 4.6 terdapat perbedaan resolusi antara penampilan data minimosd tanpa kamera dengan minimosd yang dengan kamera. Dapat dilihat pada gambar tersebut data karakter yang terdapat dibawah terpotong dan yang melebihi psosisi y akan tampil diatas atau pada posisi nol (0).

## 4.2 Pengujian Sensor

Pengujian sensor ini dimaksudakn untuk melihat hasil dari perhitungan dan proses kerja yang dilakukan oleh minimosd. Berikut akan dijelaskan beberapa pengujian sensor-sensor yang merupakan menu-menu yang dibutuhkan oleh tampilan osd.

### 4.2.1 Pengujian Baterai 5 Volt dan 12 Volt

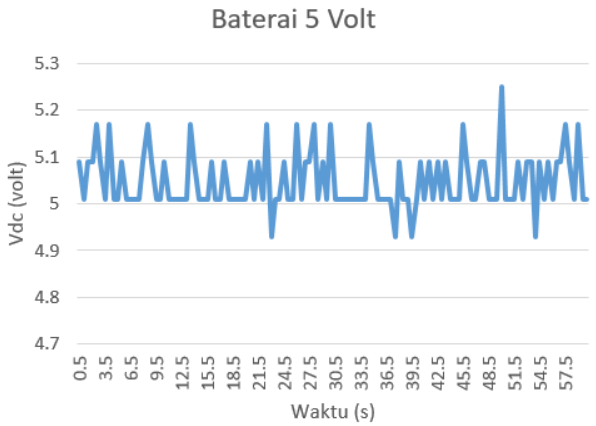
Untuk pengujian baterai dilakukan dengan du acara yaitu menguji tegangan baterai menggunakan sistem pembacaan ADC pada minimosd dan mengukur dengan sistem manual menggunakan avometer digital. Berikut pada gambar 4.7 merupakan hasil pengukuran menggunakan sistem pembacaan ADC oleh minimosd dan pada gambar 4.8 pengukuran menggunakan avometer digital.


**Gambar 4.7** Uji Coba Baterai dengan Minimosd

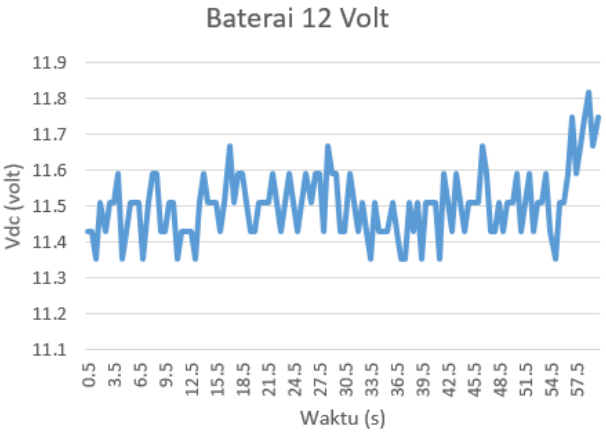

**Gambar 4.8** Uji Coba Baterai 5 Volt & 12 Volt dengan Avometer

Dari gambar diatas dapat terlihat perbedaan antara pengukuran yang dilakukan oleh minimosd dengan pembacaan ADC untuk ditampilkan pada layar monitor FPV dan dengan pengukuran yang dilakukan oleh avometer digital. Adanya perbedaan pengukuran tersebut dapat diakibatkan oleh beberapa faktor, antara lain karakteristik chip yang digunakan untuk pengukuran berbeda, keakurasian data bit ADC berbeda (pada minimosd menggunakan 10bit), adanya gangguan dari luar (misalkan dalam pengukuran menyentuh anggota tubuh), dan beberapa faktor yang lain. Selain itu, pengujian juga dilakukan dengan mengambil data tegangan yang dibaca oleh minimosd menggunakan komunikasi serial selama waktu ±1 menit. Berikut pada gambar 4.9

merupakan grafik dari hasil pengujian data baterai 5 volt tersebut. Dan pada gambar 4.10 merupakan hasil pengujian baterai 12 volt.




**Gambar 4.9** Grafik Baterai 5 Volt




**Gambar 4.10** Grafik Baterai 12 Volt

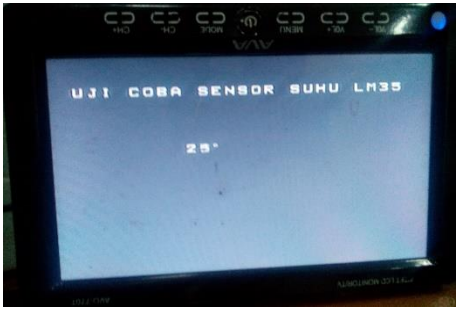
### 4.2.2 Pengujian RTC

Seperti pada tahapan-tahapan sebelumnya, pada tahapan ini dilakukan sebuah pengujian rangkaian RTC untuk melihat data kesesuaian waktu yang ditampilkan pada layar. Berikut pada gambar 4.11 merupakan hasil dari pengujian rangkaian RTC yang menggunakan modul RTC DS1307.

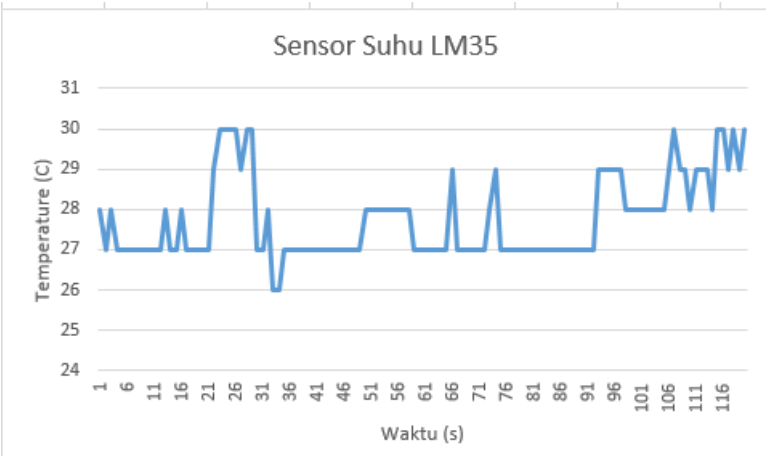

**Gambar 4.11** Uji Coba RTC

### 4.2.3 Pengujian LM35

Sensor suhu atau temperatur LM35 adalah sensor yang sangat mudah ditemukan dipasaran. Pada *mobile robot*, sensor ini berguna untuk mendeteksi suhu disekitar robot tersebut untuk mengantisipasi adanya sebuah gangguan atau adanya panas yang dapat diakibatkan oleh kerja dari robot itu sendiri yang terlalu banyak beban. Berikut pada gambar 4.12 merupakan hasil dari pengujian display sensor LM35 tersebut.




**Gambar 4.12** Uji Coba LM35



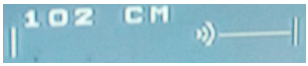

**Gambar 4.13** Grafik Sensor Suhu LM35

Sedangkan jika dilihat dari grafik yang terdapat di gambar 4.13, besar suhu yang dibaca oleh ADC minimosd dalam waktu satu menit tidak stabil, maksudnya dari hasil data yang diperoleh tidak konstan atau tidak tetap pada suhu yang sama, tetapi mengalami perubahan yang sangat drastic dengan adanya kenaikan dan penurunan besar suhu yang berkala. Hal ini dapat diakibatkan oleh pengaruh ruang pendingin yang ada diruangan seperti AC (*Air Conditioner*) yang menyala dengan mode *swing*.

## 4.3 Pengujian Serial

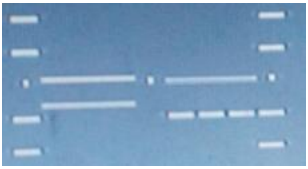
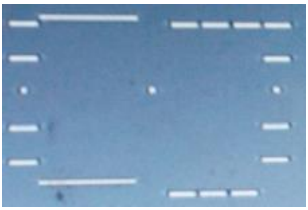
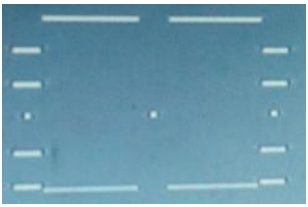
Dalam pengujian serial ini terdapat dua pengujian data yang meliputi pengujian animasi sensor ultrasonik dan pengujian posisi lengan robot. Pengujian ini dilakukan dengan perintah yang diberikan ke minimosd melalui PC (*Personal Computer*) menggunakan komunikasi serial. Pengujian ini nantinya dilakukan pada *mobile robot*, yang diintegrasikan dengan Arduino Mega untuk memperoleh data sensor ultrasonik dan posisi lengan robot. Berikut pada tabel 4.1 merupakan animasi penggambaran jarak yang dibaca oleh sesnsor ultrasonik dengan skala perbandingan 1:30.

**Tabel 4.1** Uji Sensor Ultrasonik

| Animasi | Jarak (cm) |
|---|---|
|  | 0 – 29 |
|  | 30 – 59 |
|  | 60 – 89 |
|  | 90 – 119 |
|  | 120 – 149 |
|  | 150 – 179 |
|  | 180 - 209 |
|  | 210 - 239 |
|  | 240 - 269 |
|  | 270 - 299 |
|  | >300 |

Sedangkan untuk penggambaran posisi lengan robot dilakukann dengan skala perbandingan 1:9. Berikut pada tabel 4.2 merupakan hasil dari pengskalaan tersebut.

**Tabel 4.2** Uji Posisi Lengan Robot

| Animasi | Sudut (°C) |
|---|---|
|  | 0 - 89 |
|  | 90 – 179 |
|  | 180 – 269 |
|  | >270 |

## 4.4 Pengujian APM

Pada tahap ini dilakukan pengujian data GPS oleh minimosd yang diperoleh dari ArduPilot Mega. Data GPS tersebut berfungsi untuk

informasi data keberadaan *mobile robot*. Selain data GPS yang berupa posisi robot seperti longitude dan latitude, juga terdapat data kompas atau mata angina, data kecepatan, satelit dan altitude atau ketinggian posisi robot diatas permukaan air laut. Berikut pada gambar 4.14 merupakan hasil dari pengujian tersebut.


**Gambar 4.14** Uji Coba Data APM

## 4.5 Pengujian Aplikasi pada *Mobile Robot*

Pengujian produk minimosd untuk aplikasi pada *mobile robot* ini merupakan tahapan akhir dari bab 4 tugas akhir ini. Tetapi terdapat perbedaan resolusi antara minimosd dengan kamera. Berikut merupakan hasil pengaplikasian tersebut dapat dilihat apda gambar 4.15.


**Gambar 4.15** Uji Coba Aplikasi pada *Mobile Robot*

# LAMPIRAN

## *Listing* Program Arduino

- ayodulino_osd.cpp:

```
#include <avr/io.h>
#include <avr/pgmspace.h>
#include "Arduino.h"
#include "Config.h"
#include "types.h"
#include "EEPROM.h"
#include "Max7456.h"
#include "symbols.h"
#include "Screen.h"
#include "ayodulino_osd.h"
#include "RTClib.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#if defined(ARDUINO_ARCH_SAMD)
  #define Serial SerialUSB
#endif
RTC_DS1307 rtc;
DateTime now1;
static const int RXPin = 4, TXPin = 2;
SoftwareSerial ss(RXPin, TXPin);
TinyGPSPlus gps;
uint16_t allSec=0;
uint16_t onTime=0;
uint8_t Settings[EEPROM_ITEM_LOCATION];
char screen[480];
char screenBuffer[20];
// For Home
const char beranda0[] PROGMEM = "ayodulino.tech";
const char beranda1[] PROGMEM = "Development MinimOSD";
const char beranda2[] PROGMEM = "MENU : GPS";
const char beranda3[] PROGMEM = "ARM";
const char beranda4[] PROGMEM = "BATTERY";
```

```
const char beranda5[] PROGMEM = "ULTRASONIC";
const char beranda6[] PROGMEM = "TEMEPERATURE";
const char beranda7[] PROGMEM = "msaifulhak
// For Config menu common
const char configMsgNTSC[] PROGMEM = "NTSC";
const char configMsgPAL[] PROGMEM = "PAL";
const char configMsg76[] PROGMEM = "VIDEO SYST";
// For Config Page
//-------------------------------------------------------------Page1
const char gmode0[] PROGMEM = "1/4";
const char gmode1[] PROGMEM = "GPS Mode";
const char gmode2[] PROGMEM = "hh/bb/tt";
const char gmode3[] PROGMEM = "jj:mm:dd";
const char gmode4[] PROGMEM = "Lat : ";
const char gmode5[] PROGMEM = "Lon : ";
const char gmode6[] PROGMEM = "kmph";
const char gmode7[] PROGMEM = "mdpl";
const char gmode8[] PROGMEM = "x";
const char gmode9[] PROGMEM = "GPS Tidak Terdeteksi";
//-------------------------------------------------------------Page2
const char amode0[] PROGMEM = "2/4";
const char amode1[] PROGMEM = "Armed Mode";
//-------------------------------------------------------------Page3
const char bmode0[] PROGMEM = "3/4";
const char bmode1[] PROGMEM = "Battery Mode";
const char bmode2[] PROGMEM = "V.System";
//-------------------------------------------------------------Page4
const char pmode0[] PROGMEM = "4/4";
const char pmode1[] PROGMEM = "Ping Mode";
String inputString;
boolean stringComplete = false;
String datac[30];
void parsingData(){
int i,j=0;
datac[j]="";
for(i=1;i<inputString.length();i++){
  if ((inputString[i] == '*') || (inputString[i] == ';') || (inputString[i] ==
'#'))
  {
```

```
   j++;
   datac[j]="";
  }
  else
  {
   datac[j] = datac[j] + inputString[i];
  }
}
}

void setup()
{
 Serial.begin(9600);
 ss.begin(9600);
 inputString="";

//OSD
 checkEEPROM();
 readEEPROM();
 MAX7456_Setup();
 analogReference(DEFAULT);

//RTC
 #ifndef ESP8266
 while (!Serial);
 #endif
 if (! rtc.begin()) {
  while (1);
 }
 if (! rtc.isrunning()) {
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
 }
 rtc.adjust(DateTime(2016, 5, 20, 17, 02, 0));
}

void loop()
{
 if( allSec < 10){
    displayHome(35);
```

```
    }
    else
    {
     if(stringComplete){
       parsingData();
       stringComplete=false;
       inputString="";
     }
     displayJamTgl();
     VbatOSD();
     VbatVideo();
     displaySuhu();
     displayHorizon();
     displayPing();
     JamTanggal();
     Alt();
     LatLong();
     Arah();
     Kecepatan();
     satelit();
    }
 smartDelay(500);
 MAX7456_DrawScreen();
 allSec++;
}

void serialEvent() {
  while (Serial.available()) {
    if(Serial.available()>0) {
      char inChar = (char)Serial.read();
      inputString += inChar;
      if (inChar == '#') {
        stringComplete = true;
      }
    }
  }
}
```

- ayodulino_osd.h:

```
#ifndef AYODULINO_H_
#define AYODULINO_H_
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include "RTClib.h"
#include "types.h"
      extern RTC_DS1307 rtc;
      extern DateTime now1;
      extern String datac[];
           extern SoftwareSerial ss;
           extern TinyGPSPlus gps;
           extern uint8_t Settings[];
           extern char screen[];
           extern char screenBuffer[];
           // Config status and cursor location
           extern uint8_t ROW;
           extern uint8_t COL;
           extern uint8_t configPage;
           extern uint8_t configMode;
           extern uint8_t fontMode;
           extern uint8_t fontData[];
           extern uint8_t nextCharToRequest;
           extern uint8_t lastCharToRequest;
           extern uint8_t retransmitQueue;
           // For Heading
           extern const char headGraph[] PROGMEM;
           // For Home
           extern const char beranda0[] PROGMEM;
          extern const char beranda1[] PROGMEM;
           extern const char beranda2[] PROGMEM;
           extern const char beranda3[] PROGMEM;
           extern const char beranda4[] PROGMEM;
           extern const char beranda5[] PROGMEM;
           extern const char beranda6[] PROGMEM;
            extern const char beranda7[] PROGMEM;
           // For Config menu common
           extern const char configMsgNTSC[] PROGMEM;
           extern const char configMsgPAL[] PROGMEM;
           // For Config Pages
```

```
        //------------------------------------------------------------Page1
        extern const char gmode0[] PROGMEM;
        extern const char gmode1[] PROGMEM;
        extern const char gmode2[] PROGMEM;
        extern const char gmode3[] PROGMEM;
        extern const char gmode4[] PROGMEM;
        extern const char gmode5[] PROGMEM;
        extern const char gmode6[] PROGMEM;
        extern const char gmode7[] PROGMEM;
        extern const char gmode8[] PROGMEM;
         extern const char gmode9[] PROGMEM;
        //------------------------------------------------------------Page2
        extern const char amode0[] PROGMEM;
        extern const char amode1[] PROGMEM;
        //------------------------------------------------------------Page3
        extern const char bmode0[] PROGMEM;
        extern const char bmode1[] PROGMEM;
        extern const char bmode2[] PROGMEM;
        extern const char bmode3[] PROGMEM;
        //------------------------------------------------------------Page4
        extern const char pmode0[] PROGMEM;
        extern const char pmode1[] PROGMEM;
        extern const char configMsg76[] PROGMEM;
                // Variables for items pos change on screen
        //------------------------------------------------------------
        extern int8_t screenitemselect; // pointer for item text strings
        extern int8_t screen_pos_item_pointer;
#endif
```

- screen.cpp:

```
#include "Arduino.h"
#include "Config.h"
#include "Max7456.h"
#include "symbols.h"
#include "ayodulino_osd.h"
#include "Screen.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <stdlib.h>
```

```
void displayHome(char position)
{
  MAX7456_WriteString_P(beranda0, position);
  MAX7456_WriteString_P(configMsg76, position+LINE);   // "VIDEO
SYSTEM"
  if (Settings[S_VIDEOSIGNALTYPE])
    MAX7456_WriteString_P(configMsgPAL, position+43);  // PAL
  else
    MAX7456_WriteString_P(configMsgNTSC, position+43);  // NTSC
  MAX7456_WriteString_P(beranda1, position+120+LINE);
  MAX7456_WriteString_P(beranda2,
position+120+LINE+LINE+LINE);
  MAX7456_WriteString_P(beranda3,
position+127+LINE+LINE+LINE+LINE);
  MAX7456_WriteString_P(beranda4,
position+127+LINE+LINE+LINE+LINE+LINE);
  MAX7456_WriteString_P(beranda5,
position+127+LINE+LINE+LINE+LINE+LINE+LINE);
  MAX7456_WriteString_P(beranda6,
position+127+LINE+LINE+LINE+LINE+LINE+LINE+LINE);
  MAX7456_WriteString_P(beranda7,
position+120+LINE+LINE+LINE+LINE+LINE+LINE+LINE+LINE+L
INE);
}

void displayJamTgl(void){
 now1 = rtc.now();
 char h[4],bb[4],t[8],j[4],m[4],dd[4];
 char gar[2] = "/";
 char tit[2] = ":";
 char nol[2] = "0";
 String a = String(now1.day());
 a.toCharArray(h,3);
 String b = String(now1.month());
 b.toCharArray(bb,3);
 String c = String(now1.year());
 c.toCharArray(t,5);
 String d = String(now1.hour());
```

```
 d.toCharArray(j,3);
 String e = String(now1.minute());
 e.toCharArray(m,3);
 String f = String(now1.second());
 f.toCharArray(dd,3);
// Serial.print(now1.year(), DEC);
 if (a.toInt()<10){
   MAX7456_WriteString(nol,19);
   MAX7456_WriteString(h,20);
 }
 else {
   MAX7456_WriteString(h,19);
 }
 if (b.toInt()<10){
   MAX7456_WriteString(nol,22);
   MAX7456_WriteString(bb,23);
 }
 else {
   MAX7456_WriteString(bb,22);
 }
 if (d.toInt()<10){
   MAX7456_WriteString(nol,51);
   MAX7456_WriteString(j,52);
 }
 else {
   MAX7456_WriteString(j,51);
 }
 if (e.toInt()<10){
   MAX7456_WriteString(nol,54);
   MAX7456_WriteString(m,55);
 }
 else {
    MAX7456_WriteString(m,54);
 }
 if (f.toInt()<10){
   MAX7456_WriteString(nol,57);
   MAX7456_WriteString(dd,58);
 }
 else {
```

```
   MAX7456_WriteString(dd,57);
 }
 MAX7456_WriteString(t,25);
 MAX7456_WriteString(tit,53);
 MAX7456_WriteString(tit,56);
 MAX7456_WriteString(gar,21);
 MAX7456_WriteString(gar,24);
}

void VbatOSD(void)
{
  int VDig5;
  float VAna5;
  char disp[5];
  float LevBat0 = 2.5;
  float LevBat1 = 3;
  float LevBat2 = 3.5;
  float LevBat3 = 4;
  float LevBat4 = 4.5;
  float LevBat5 = 5;
  VDig5 = analogRead(VbatOSDPin);
  VAna5 = (float)VDig5* 0.0783;
  dtostrf(VAna5,2,2,disp);
  MAX7456_WriteString(disp,413);
  screenBuffer[0] = 0x8e;
  MAX7456_WriteString(screenBuffer,418);
      if (VAna5 < LevBat0) screenBuffer[0] = 0xb4;
  else if (VAna5 < LevBat1) screenBuffer[0] = 0xb5;
  else if (VAna5 < LevBat2) screenBuffer[0] = 0xb6;
  else if (VAna5 < LevBat3) screenBuffer[0] = 0xb7;
  else if (VAna5 < LevBat4) screenBuffer[0] = 0xb8;
  else if (VAna5 < LevBat5) screenBuffer[0] = 0xb9;
  else                      screenBuffer[0] = 0xba;          // Max charge icon
  MAX7456_WriteString(screenBuffer,412);
}

void VbatVideo(void)
{
  int VDig12;
```

```
  float VAna12;
  char disp[5];
  float LevBat0 = 9;
  float LevBat1 = 9.5;
  float LevBat2 = 10;
  float LevBat3 = 10.5;
  float LevBat4 = 11;
  float LevBat5 = 11.5;
  VDig12 = analogRead(VbatVideoPin);
  VAna12 = (float)VDig12 * 0.0783;
  dtostrf(VAna12,2,2,disp);
  MAX7456_WriteString(disp,443);
  screenBuffer[0] = 0x8e;
  MAX7456_WriteString(screenBuffer,448);
      if (VAna12 < LevBat0) screenBuffer[0] = 0xb4;
  else if (VAna12 < LevBat1) screenBuffer[0] = 0xb5;
  else if (VAna12 < LevBat2) screenBuffer[0] = 0xb6;
  else if (VAna12 < LevBat3) screenBuffer[0] = 0xb7;
  else if (VAna12 < LevBat4) screenBuffer[0] = 0xb8;
  else if (VAna12 < LevBat5) screenBuffer[0] = 0xb9;
  else                  screenBuffer[0] = 0xba;            // Max charge icon
 MAX7456_WriteString(screenBuffer,442);
}

void displaySuhu(void)
{
  int VDsuhu;
  float derSuhu;
  char disp[3];
  VDsuhu = analogRead(suhuPin);
  derSuhu = (float)VDsuhu * 0.4888;
  dtostrf(derSuhu,2,0,disp);
  MAX7456_WriteString(disp,386);
  screenBuffer[0] = 0xb0;
  screenBuffer[1] = 0;
  MAX7456_WriteString(screenBuffer,388);
}

void displayHorizon(void)
```

```
{
 int teta1,teta2,teta3,teta4,a1,a2,a3,a4,i;
 char buf[4];
 screen[195]=0x2e;
 screen[131]=5;
 screen[161]=5;
 screen[191]=0x2e;
 screen[221]=5;
 screen[251]=5;
 screen[139]=5;
 screen[169]=5;
 screen[199]=0x2e;
 screen[229]=5;
 screen[259]=5;
 teta1 = datac[1].toInt();
 teta2 = datac[2].toInt();
 teta3 = datac[3].toInt();
 teta4 = datac[4].toInt();

 if (teta1<0){
   teta1 = teta1*(-1);
 }
 if (teta2<0){
   teta2 = teta2*(-1);
 }
 if (teta3<0){
   teta3 = teta3*(-1);
 }
 if (teta4<0){
   teta4 = teta4*(-1);
 }

 for(i=0;i<3;i++){
  if (teta1>=0 && teta1<90){
    a1 = teta1/9;
    screen[196+i]=5+a1;
  }
  if (teta1>=90 && teta1<180){
    a1 = (teta1-90)/9;
```

```
    screen[166+i]=5+a1;
   }
   if (teta1>=180 && teta1<270){
    a1 = (teta1-180)/9;
    screen[136+i]=5+a1;
   }
 }

for(i=0;i<3;i++){
   if (teta2>=0 && teta2<90){
    a1 = teta2/9;
    screen[192+i]=5+a1;
   }
   if (teta2>=90 && teta2<180){
    a1 = (teta2-90)/9;
    screen[162+i]=5+a1;
   }
   if (teta2>=180 && teta2<270){
    a1 = (teta2-180)/9;
    screen[132+i]=5+a1;
   }
 }

 for(i=0;i<3;i++){
   if (teta3>=0 && teta3<90){
    a1 = teta3/9;
    screen[222+i]=14-a1;
   }
   if (teta3>=90 && teta3<180){
    a1 = (teta3-90)/9;
    screen[252+i]=14-a1;
   }
   if (teta3>=180 && teta3<270){
    a1 = (teta3-180)/9;
    screen[282+i]=14-a1;
   }
 }

 for(i=0;i<3;i++){
```

```
    if (teta4>=0 && teta4<90){
      a1 = teta4/9;
      screen[226+i]=14-a1;
    }
    if (teta4>=90 && teta4<180){
      a1 = (teta4-90)/9;
      screen[256+i]=14-a1;
    }
    if (teta4>=180 && teta4<270){
      a1 = (teta4-180)/9;
      screen[286+i]=14-a1;
    }
  }
}

void displayPing(void)
{
  int a,b;
  char buf[4];
  a = datac[0].toInt();
  a = a/30;

  if (datac[0].toInt()>300){
   a=10;
   char disp[4] = "max";
   MAX7456_WriteString(disp,318);
  }
  if (datac[0].toInt()<30){
   char disp[4] = "min";
   MAX7456_WriteString(disp,318);
  }
  if (datac[0].toInt()>0){
    int i = datac[0].length();
    char disp[4] = "cm";
    MAX7456_WriteString(disp,310+i+1);
  }
  screen[350-a]=0xfa;
  screen[339]=0xd0;
  screen[351]=0xd1;
```

```
  for(b=1;b<=a;b++){
    screen[350-b+1]=0xc0;
  }
  datac[0].toCharArray(buf, 4);
  MAX7456_WriteString(buf,310);
}

void noGPS(void)
{
if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    MAX7456_WriteString_P(gmode9, 185);
    while(true);
  }
}

void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    while (ss.available())
      gps.encode(ss.read());
  } while (millis() - start < ms);
}

void displayGPS(void)
{
  MAX7456_WriteString_P(gmode0, 1);
  MAX7456_WriteString_P(gmode1, 31);
  smartDelay(0);
}

void JamTanggal(void)
{
  int a,b;
  TinyGPSDate d = gps.date;
  TinyGPSTime t = gps.time;
```

```
 if (t.isValid())
 {
  char disp[10];
  a = t.hour();
  if ((a+7) > 23){
    b = 1;
   sprintf(disp, "%02d:%02d:%02d ", a+7-24, t.minute(), t.second());
  }
  else{
    b = 0;
   sprintf(disp, "%02d:%02d:%02d ", a+7, t.minute(), t.second());
  }
  MAX7456_WriteString(disp,51);
 }
 else
 {
  MAX7456_WriteString_P(gmode3, 51);
 }

 if (d.isValid())
 {
  char disp[10];
  if (b==0){
   sprintf(disp, "%02d/%02d/%02d ",  d.day(), d.month(), d.year());
  }
  else{
   sprintf(disp, "%02d/%02d/%02d ",  d.day()+1, d.month(), d.year());
  }
  MAX7456_WriteString(disp,19);
 }
 else
 {
  MAX7456_WriteString_P(gmode2, 21);
 }
}

void satelit(void)
{
 screenBuffer[0] = 0X0f;
```

```
  MAX7456_WriteString(screenBuffer,31);
  if (gps.satellites.isValid())
  {
    char disp[32];
    dtostrf(gps.satellites.value(),2,0,disp);
    MAX7456_WriteString(disp,32);
  }
  else
  {
    MAX7456_WriteString_P(gmode8, 33);
  }
  smartDelay(0);
}

void LatLong(void)
{
  screenBuffer[0]=0X83;
  MAX7456_WriteString(screenBuffer,391);
  screenBuffer[0]=0X84;
  MAX7456_WriteString(screenBuffer,421);
  if (gps.location.isValid())
  {
    char disp[32];
    dtostrf( gps.location.lat(),3,5,disp);
    MAX7456_WriteString(disp,393);
    dtostrf(gps.location.lng(),3,5,disp);
    MAX7456_WriteString(disp,423);
  }
  else
  {
    MAX7456_WriteString_P(gmode8, 393);
    MAX7456_WriteString_P(gmode8, 423);
  }
  smartDelay(0);
}

void Arah(void)
{
  screenBuffer[0] = 0xae;
```

```
  screenBuffer[1] = 0xaf;
  MAX7456_WriteString(screenBuffer,46);
  screenBuffer[0] = 0xb0;
  screenBuffer[1] = 0;
  MAX7456_WriteString(screenBuffer,45);
  if (gps.course.isValid())
  {
    char disp[32];
    dtostrf(gps.course.deg(),3,2,disp);
    MAX7456_WriteString(disp,39);
  }
  else
  {
    MAX7456_WriteString_P(gmode8, 44);
  }
  smartDelay(0);
}

void Kecepatan(void)
{
  screenBuffer[0]=0X81;
  MAX7456_WriteString(screenBuffer,181);
  if (gps.speed.isValid())
  {
    char disp[32];
    dtostrf(gps.speed.kmph(),3,2,disp);
    MAX7456_WriteString(disp,183);
  }
  else
  {
    MAX7456_WriteString_P(gmode8, 183);
  }
  smartDelay(0);
}

void Alt(void)
{
  screenBuffer[0]=0X85;
  MAX7456_WriteString(screenBuffer,361);
```

```
  if (gps.altitude.isValid())
  {
    char disp[32];
    dtostrf(gps.altitude.meters(),3,2,disp);
    MAX7456_WriteString(disp,363);
  }
  else
  {
    MAX7456_WriteString_P(gmode8, 363);
  }
  smartDelay(0);
}

void displayArm(void)
{
  MAX7456_WriteString_P(amode0, 1);
  MAX7456_WriteString_P(amode1, 31);
  smartDelay(0);
}

void displayFontScreen(void) {
  MAX7456_WriteString_P(PSTR("FONT MAX7456"), 38);

  for(uint16_t i = 0; i < 256; i++)
    screen[90+i] = i;
}

void arm1(void)
{
  int VDArm;
  float VAArm;
  char disp[32];
  VDArm = analogRead(armPin1);
  VAArm = (float)VDArm * 5 / 1023;

  dtostrf(VAArm,2,2,disp);
  MAX7456_WriteString(disp,403);
}
```

```
void displayBatt(void)
{
  MAX7456_WriteString_P(bmode0, 1);
  MAX7456_WriteString_P(bmode1, 31);
  smartDelay(0);
}

char *formatTime(uint16_t val, char *str, uint8_t hhmmss) {
  int8_t bytes = 5;
  if(hhmmss)
    bytes = 8;
  str[bytes] = 0;
  do {
    str[--bytes] = '0' + (val % 10);
    val = val / 10;
    str[--bytes] = '0' + (val % 6);
    val = val / 6;
    str[--bytes] = ':';
  } while(hhmmss-- != 0);
  do {
    str[--bytes] = '0' + (val % 10);
    val = val / 10;
  } while(val != 0 && bytes != 0);
  while(bytes != 0)
     str[--bytes] = ' ';
  return str;
}

void displayTimer(void)
{
  screenBuffer[0] = SYM_ON;
  formatTime(onTime,                screenBuffer+1,                0);
MAX7456_WriteString(screenBuffer,((Settings[L_ONTIMEPOSITION
ROW]-1)*30) + Settings[L_ONTIMEPOSITIONCOL]);
   }
  }
}
```

- screen.h:

```
#ifndef SCREEN_H_
#define SCREEN_H_
#include "Arduino.h"
      void displayHome(char position);
      void displayHorizon(void);
      void displayPing(void);
      void noGPS(void);
      void smartDelay(unsigned long ms);
      void displayGPS(void);
      void JamTanggal(void);
      void LatLong(void);
      void Arah(void);
      void Kecepatan(void);
      void Alt(void);
      void satelit(void);
      void displayArm(void);
      void arm1(void);
      void displayBatt(void);
      void VbatVideo(void);
      void VbatOSD(void);
      void displayMPU(void);
      void displayJamTgl(void);
      void displaySuhu(void);
      void displayFontScreen(void);
#endif
```

- Max7456.cpp:

```
#include "Arduino.h"
#include "ayodulino_osd.h"
#include "Max7456.h"
// Selectable by board type
uint8_t MAX7456SELECT;              // output pin
uint8_t MAX7456RESET;               // output pin
// Selectable by video mode
uint16_t MAX_screen_size;
uint8_t MAX7456_reset;
uint8_t MAX_screen_rows;
```

```
volatile uint8_t vsync_wait = 0;
uint8_t spi_transfer(uint8_t data)
{
 SPDR = data;                    // Start the transmission
 while (!(SPSR & (1<<SPIF)))     // Wait the end of the transmission
   ;
 return SPDR;                    // return the received byte
}

void MAX7456_Send(uint8_t add, uint8_t data)
{
        spi_transfer(add);
        spi_transfer(data);
}
ISR(INT0_vect) {
        vsync_wait = 0;
}

void MAX7456Configure() {
        // todo - automatically recognising card.
        if(Settings[S_BOARDTYPE] == 0) {        // Rush
                MAX7456SELECT = 10;     // ss
                MAX7456RESET  = 9;      // RESET
        }
        if(Settings[S_BOARDTYPE] == 1) {        // MinimOSD
                MAX7456SELECT = 6;      // ss
                MAX7456RESET = 10;      // RESET
        }
}

void MAX7456_Setup(void)
{
 MAX7456Configure();
 if(Settings[S_VIDEOSIGNALTYPE]) {    // PAL
   MAX7456_reset = 0x42;
   MAX_screen_size = 480;
   MAX_screen_rows = 16;
 }
 else {                          // NTSC
```

```
  MAX7456_reset = 0x02;
  MAX_screen_size = 390;
  MAX_screen_rows = 13;
}
pinMode(MAX7456RESET,OUTPUT);
digitalWrite(MAX7456RESET,HIGH); //hard enable
delay(250);
pinMode(MAX7456SELECT,OUTPUT);
digitalWrite(MAX7456SELECT,HIGH); //disable device
pinMode(DATAOUT, OUTPUT);
pinMode(DATAIN, INPUT);
pinMode(SPICLOCK,OUTPUT);
pinMode(VSYNC, INPUT);
// SPCR = 01010000
//interrupt disabled,spi enabled,msb 1st,master,clk low when idle,
//sample on leading edge of clk,system clock/4 rate (4 meg)
SPCR = (1<<SPE)|(1<<MSTR);
SPSR=(1<<SPI2X);
uint8_t spi_junk;
spi_junk=SPSR;
spi_junk=SPDR;
delay(250);
// force soft reset on Max7456
digitalWrite(MAX7456SELECT,LOW);
MAX7456_Send(VM0_reg, MAX7456_reset);
digitalWrite(MAX7456SELECT,HIGH);
delay(500);
// set all rows to same character white level, 90%
digitalWrite(MAX7456SELECT,LOW);
uint8_t x;
for(x = 0; x < MAX_screen_rows; x++) {
  MAX7456_Send(MAX7456ADD_RB0+x, WHITE_level_120);
}
// make sure the Max7456 is enabled
spi_transfer(VM0_reg);
if (Settings[S_VIDEOSIGNALTYPE]){
  spi_transfer(OSD_ENABLE|VIDEO_MODE_PAL);
}
else{
```

```
   spi_transfer(OSD_ENABLE|VIDEO_MODE_NTSC);
  }
  digitalWrite(MAX7456SELECT,HIGH);
  delay(100);
  EIMSK |= (1 << INT0);  // enable interuppt
  EICRA |= (1 << ISC01); // interrupt at the falling edge
  sei();
}

// Copy string from ram into screen buffer
void MAX7456_WriteString(const char *string, int Adresse)
{
        uint8_t xx;
        for(xx=0;string[xx]!=0;)
        {
                screen[Adresse++] = string[xx++];
        }
}

// Copy string from progmem into the screen buffer
void MAX7456_WriteString_P(const char *string, int Adresse)
{
        uint8_t xx = 0;
        char c;
        while((c = (char)pgm_read_byte(&string[xx++])) != 0)
        {
                screen[Adresse++] = c;
        }
}

void MAX7456_writeNVM(uint8_t char_address)
{
        #ifdef WRITE_TO_MAX7456
        // disable display
        digitalWrite(MAX7456SELECT,LOW);
        spi_transfer(VM0_reg);
        spi_transfer(Settings[S_VIDEOSIGNALTYPE]?0x40:0);
        spi_transfer(MAX7456ADD_CMAH); // set start address high
        spi_transfer(char_address);
```

```
        for(uint8_t x = 0; x < NVM_ram_size; x++) // write out 54 bytes of character to shadow ram
        {
                spi_transfer(MAX7456ADD_CMAL); // set start address low
                spi_transfer(x);
                spi_transfer(MAX7456ADD_CMDI);
                spi_transfer(fontData[x]);
        }
        // transfer 54 bytes from shadow ram to NVM
        spi_transfer(MAX7456ADD_CMM);
        spi_transfer(WRITE_nvr);
        // wait until bit 5 in the status register returns to 0 (12ms)
        while ((spi_transfer(MAX7456ADD_STAT) & STATUS_reg_nvr_busy) != 0x00);
        spi_transfer(VM0_reg); // turn on screen next vertical
        spi_transfer(Settings[S_VIDEOSIGNALTYPE]?0x4c:0x0c);
        digitalWrite(MAX7456SELECT,HIGH);
        #else
        delay(12);
        #endif
}

void MAX7456_DrawScreen()
{
 vsync_wait = 1;
 while (vsync_wait) ;
 int xx;
 digitalWrite(MAX7456SELECT,LOW);
 spi_transfer(DMM_reg);
 spi_transfer(1);
 spi_transfer(DMAH_reg);
 spi_transfer(0);
 spi_transfer(DMAL_reg);
 spi_transfer(0);
 for(xx=0;xx<MAX_screen_size;++xx)
 {
  MAX7456_Send(MAX7456ADD_DMDI, screen[xx]);
```

```
   screen[xx] = ' ';
  }
  spi_transfer(DMDI_reg);
  spi_transfer(END_string);
  spi_transfer(DMM_reg);
  spi_transfer(B00000000);
  digitalWrite(MAX7456SELECT,HIGH);
}
```

- EEPROM.cpp:

```
#include <avr/eeprom.h>
#include "Arduino.h"
#include "EEPROM.h"
#include "types.h"
#include "ayodulino_osd.h"


uint8_t EEPROMClass::read(int address)
{
        return eeprom_read_byte((unsigned char *) address);
}

void EEPROMClass::write(int address, uint8_t value)
{
        eeprom_write_byte((unsigned char *) address, value);
}

EEPROMClass EEPROM;
// For Settings Defaults
static const uint8_t EEPROM_DEFAULT[EEPROM_SETTINGS] = {
1,   // used for check          0
0,   // S_RSSIMIN               1
255, // S_RSSIMAX               2
60,  // S_RSSI_ALARM            3
1,   // S_MWRSSI                4
0,   // S_PWMRSSI               5
8,    // S_PWMRSSIDIVIDER              6     // PWM Freq 500Hz=8,
1KHz=4 (Divider to avoid value >255)
105, // S_VOLTAGEMIN            7
```

3, // S_BATCELLS 8
100, // S_DIVIDERRATIO 9
1, // S_MAINVOLTAGE_VBAT 10
100, // S_VIDDIVIDERRATIO 11
0, // S_VIDVOLTAGE_VBAT 12
//90, // S_TEMPERATUREMAX 13 // Do not remove yet
1, // S_BOARDTYPE 14
1, // S_DISPLAYGPS 15
1, // S_COORDINATES 16
0, // S_HEADING360 17
0, // S_UNITSYSTEM 18
1, // S_VIDEOSIGNALTYPE 19
0, // S_RESETSTATISTICS 20
1, // S_ENABLEADC 21
5, // S_BLINKINGHZ 22 // 10=1Hz, 9=1.1Hz, 8=1,25Hz, 7=1.4Hz, 6=1.6Hz, 5=2Hz, 4=2,5Hz, 3=3,3Hz, 2=5Hz, 1=10Hz
0, // S_MWAMPERAGE 23
40, // S_CURRSENSSENSITIVITY 24 // May vary from 17 to 40mV/A (Sensor type)
2, // S_CURRSENSOFFSET_H 25 // offset(H/L) =0 for unidir sensors or =512 for bidirectional sensors, may be changed only of few units.
0, // S_CURRSENSOFFSET_L 26 // 2H+0L=512
2, // S_CLIMB_RATE_ALARM 27
5, // S_VOLUME_DIST_MAX 28 // Flying Volume Warning (Distance value in meters x100) by default is 500m
25, // S_VOLUME_ALT_MAX 29 // " " " (Altitude Max " " " x2 ) " " " 50m
0, // S_VOLUE_ALT_MIN 30 // " " " (Altitude Min " " " ___ " " " 0m
105, // S_VIDVOLTAGEMIN 31
30, // S_PITCH_WARNING 32 // Warning message at given angle in degrees positive and negative (default 30)

0, //S_CALLSIGN 32 // TEXT CONFIGURATION ONLY (On by default using L_CALLSIGNPOSITIONDSPL)
0, // S_CS0, 33 // 10 callsign char locations
0, // S_CS1,
0, // S_CS2,

```
0,   // S_CS3,
0,   // S_CS4,
0,   // S_CS5,
0,   // S_CS6,
0,   // S_CS7,
0,   // S_CS8,
0,   // S_CS9,              42
};
// PAL item position Defaults
static                          const                          uint8_t
EEPROM_PAL_DEFAULT[EEPROM_ITEM_LOCATION-
EEPROM_SETTINGS] = {
        // ROW= Row position on screen (255= no action)
        // COL= Column position on screen (255= no action)
        // DSPL= Display item on screen

        2,   // L_GPS_NUMSATPOSITIONROW LINE02+6
        18,  // L_GPS_NUMSATPOSITIONCOL
        1,   // L_GPS_NUMSATPOSITIONDSPL

        6,   // L_GPS_DIRECTIONTOHOMEPOSROW LINE03+14
        2,   // L_GPS_DIRECTIONTOHOMEPOSCOL
        1,   // L_GPS_DIRECTIONTOHOMEPOSDSPL

        10,   // L_GPS_DISTANCETOHOMEPOSROW LINE02+24
        2,   // L_GPS_DISTANCETOHOMEPOSCOL
        1,   // L_GPS_DISTANCETOHOMEPOSDSPL

        10,  // L_SPEEDPOSITIONROW LINE03+24
        23,   // L_SPEEDPOSITIONCOL
        1,   // L_SPEEDPOSITIONDSPL

        9,   // L_GPS_ANGLETOHOMEPOSROW LINE04+12
        2,   // L_GPS_ANGLETOHOMEPOSCOL
        0,   // L_GPS_ANGLETOHOMEPOSDSPL

        /*13,   // L_MW_GPS_ALTPOSITIONROW LINE04+24    Do
not remove yet
        2,   // L_MW_GPS_ALTPOSITIONCOL
```

```
0,   // L_MW_GPS_ALTPOSITIONDSPL*/
2,   // L_SENSORPOSITIONROW LINE03+2
24,  // L_SENSORPOSITIONCOL
1,   // L_SENSORPOSITIONDSPL
2,   // L_MODEPOSITIONROW   LINE05+2
8,   // L_MODEPOSITIONCOL
1,   // L_MODEPOSITIONDSPL
3,   // L_MW_HEADINGPOSITIONROW LINE02+19
2,   // L_MW_HEADINGPOSITIONCOL
1,   // L_MW_HEADINGPOSITIONDSPL
2,   // L_MW_HEADINGGRAPHPOSROW LINE02+10
2,   // L_MW_HEADINGGRAPHPOSCOL
1,   // L_MW_HEADINGGRAPHPOSDSPL

/*12,   // L_TEMPERATUREPOSROW  LINE11+2   // Do not remove yet
2,   // L_TEMPERATUREPOSCOL
0,   // L_TEMPERATUREPOSDSPL*/
6,  // L_MW_ALTITUDEPOSITIONROW LINE08+2
23,  // L_MW_ALTITUDEPOSITIONCOL
1,   // L_MW_ALTITUDEPOSITIONDSPL
8,   // L_CLIMBRATEPOSITIONROW LINE08+24
5,  // L_CLIMBRATEPOSITIONCOL
1,   // L_CLIMBRATEPOSITIONDSPL
6,   // L_HORIZONPOSITIONROW LINE06+8
8,   // L_HORIZONPOSITIONCOL
1,   // L_HORIZONPOSITIONDSPL
255, // L_HORIZONSIDEREFROW
255, // L_HORIZONSIDEREFCOL
1,   // L_HORIZONSIDEREFDSPL
255, // L_HORIZONCENTERREFROW
255, // L_HORIZONCENTERREFCOL
1,   // L_HORIZONCENTERREFDSPL
7,   // L_CURRENTTHROTTLEPOSITIONROW LINE14+22
20,  // L_CURRENTTHROTTLEPOSITIONCOL
1,   // L_CURRENTTHROTTLEPOSITIONDSPL
15,   // L_FLYTIMEPOSITIONROW LINE15+22
14,   // L_FLYTIMEPOSITIONCOL
1,    // L_FLYTIMEPOSITIONDSPL
```

```
        15,   // L_ONTIMEPOSITIONROW LINE15+22
        14,   // L_ONTIMEPOSITIONCOL
        1,    // L_ONTIMEPOSITIONDSPL
        3,   // L_MOTORARMEDPOSITIONROW LINE14+11
        24,  // L_MOTORARMEDPOSITIONCOL
        1,   // L_MOTORARMEDPOSITIONDSPL
        14,   // L_MW_GPS_LATPOSITIONROW  LINE12+2
        2,    // L_MW_GPS_LATPOSITIONCOL
        1,    // L_MW_GPS_LATPOSITIONDSPL
        15,   // L_MW_GPS_LONPOSITIONROW  LINE12+15
        2,    // L_MW_GPS_LONPOSITIONCOL
        1,    // L_MW_GPS_LONPOSITIONDSPL
        2,    / L_RSSIPOSITIONROW LINE14+2
        12,   // L_RSSIPOSITIONCOL
        1,    // L_RSSIPOSITIONDSPL
        15,   // L_VOLTAGEPOSITIONROW LINE15+3
        23,   // L_VOLTAGEPOSITIONCOL
        1,    // L_VOLTAGEPOSITIONDSPL
        255,  // L_MAINBATLEVEVOLUTIONROW,
        255,  // L_MAINBATLEVEVOLUTIONCOL,
        1,    // L_MAINBATLEVEVOLUTIONDSPL,
        13,   // L_VIDVOLTAGEPOSITIONROW LINE13+3
        23,   // L_VIDVOLTAGEPOSITIONCOL
        0,    // L_VIDVOLTAGEPOSITIONDSPL
        14,   // L_AMPERAGEPOSITIONROW LINE15+10
        23,   // L_AMPERAGEPOSITIONCOL
        1,    // L_AMPERAGEPOSITIONDSPL
        14,   // L_PMETERSUMPOSITIONROW LINE15+16
        14,   // L_PMETERSUMPOSITIONCOL
        1,   // L_PMETERSUMPOSITIONDSPL
        13,   // L_CALLSIGNPOSITIONROW LINE14+10
        10,   // L_CALLSIGNPOSITIONCOL
        1,    // L_CALLSIGNPOSITIONDSPL
};

// NTSC item position Defaults
uint8_t   EEPROM_NTSC_DEFAULT[EEPROM_ITEM_LOCATION-
EEPROM_SETTINGS] = {
        // ROW= Row position on screen (255= no action)
```

```
        // COL= Column position on screen (255= no action)
        // DSPL= Display item on screen
        2,   // L_GPS_NUMSATPOSITIONROW LINE02+6
        18,  // L_GPS_NUMSATPOSITIONCOL
        1,   // L_GPS_NUMSATPOSITIONDSPL
        6,   // L_GPS_DIRECTIONTOHOMEPOSROW LINE03+14
        2,   // L_GPS_DIRECTIONTOHOMEPOSCOL
        1,   // L_GPS_DIRECTIONTOHOMEPOSDSPL
        10,   // L_GPS_DISTANCETOHOMEPOSROW LINE02+24
        2,   // L_GPS_DISTANCETOHOMEPOSCOL
        1,   // L_GPS_DISTANCETOHOMEPOSDSPL
        10,  // L_SPEEDPOSITIONROW LINE03+24
        23,   // L_SPEEDPOSITIONCOL
        1,   // L_SPEEDPOSITIONDSPL
        9,   // L_GPS_ANGLETOHOMEPOSROW LINE04+12
        2,   // L_GPS_ANGLETOHOMEPOSCOL
        0,   // L_GPS_ANGLETOHOMEPOSDSPL

        /*11,  // L_MW_GPS_ALTPOSITIONROW LINE04+24    Do
not remove yet
        2,   // L_MW_GPS_ALTPOSITIONCOL
        0,   // L_MW_GPS_ALTPOSITIONDSPL*/
        2,   // L_SENSORPOSITIONROW LINE03+2
        24,  // L_SENSORPOSITIONCOL
        1,   // L_SENSORPOSITIONDSPL
        2,   // L_MODEPOSITIONROW   LINE05+2
        8,  // L_MODEPOSITIONCOL
        1,   // L_MODEPOSITIONDSPL
        3,   // L_MW_HEADINGPOSITIONROW LINE02+19
        2,   // L_MW_HEADINGPOSITIONCOL
        1,   // L_MW_HEADINGPOSITIONDSPL
        2,   // L_MW_HEADINGGRAPHPOSROW LINE02+10
        2,   // L_MW_HEADINGGRAPHPOSCOL
        1,   // L_MW_HEADINGGRAPHPOSDSPL
        /*12,  // L_TEMPERATUREPOSROW LINE11+2         // Do
not remove yet
        2,   // L_TEMPERATUREPOSCOL
        0,   // L_TEMPERATUREPOSDSPL*/
        6,  // L_MW_ALTITUDEPOSITIONROW LINE08+2
```

```
23,  // L_MW_ALTITUDEPOSITIONCOL
1,   // L_MW_ALTITUDEPOSITIONDSPL
8,   // L_CLIMBRATEPOSITIONROW LINE08+24
5,  // L_CLIMBRATEPOSITIONCOL
1,   // L_CLIMBRATEPOSITIONDSPL
6,   // L_HORIZONPOSITIONROW LINE06+8
8,   // L_HORIZONPOSITIONCOL
1,   // L_HORIZONPOSITIONDSPL
255, // L_HORIZONSIDEREFROW
255, // L_HORIZONSIDEREFCOL
1,   // L_HORIZONSIDEREFDSPL
255, // L_HORIZONCENTERREFROW
255, // L_HORIZONCENTERREFCOL
1,   // L_HORIZONCENTERREFDSP
7,   // L_CURRENTTHROTTLEPOSITIONROW LINE14+22
20,  // L_CURRENTTHROTTLEPOSITIONCOL
1,   // L_CURRENTTHROTTLEPOSITIONDSPL
13,   // L_FLYTIMEPOSITIONROW LINE15+22
14,   // L_FLYTIMEPOSITIONCOL
1,    // L_FLYTIMEPOSITIONDSPL
13,   // L_ONTIMEPOSITIONROW LINE15+22
14,   // L_ONTIMEPOSITIONCOL
1,    // L_ONTIMEPOSITIONDSPL
3,    // L_MOTORARMEDPOSITIONROW LINE14+11
24,   // L_MOTORARMEDPOSITIONCOL
1,    // L_MOTORARMEDPOSITIONDSPL
12,   // L_MW_GPS_LATPOSITIONROW  LINE12+2
2,    // L_MW_GPS_LATPOSITIONCOL
1,    // L_MW_GPS_LATPOSITIONDSPL
13,   // L_MW_GPS_LONPOSITIONROW  LINE12+15
2,    // L_MW_GPS_LONPOSITIONCOL
1,    // L_MW_GPS_LONPOSITIONDSPL
2,    // L_RSSIPOSITIONROW LINE14+2
12,   // L_RSSIPOSITIONCOL
1,    // L_RSSIPOSITIONDSPL
13,   // L_VOLTAGEPOSITIONROW LINE15+3
23,   // L_VOLTAGEPOSITIONCOL
1,    // L_VOLTAGEPOSITIONDSPL
255,  // L_MAINBATLEVEVOLUTIONROW
```

```
        255,  // L_MAINBATLEVEVOLUTIONCOL
        1,    // L_MAINBATLEVEVOLUTIONDSPL
        11,   // L_VIDVOLTAGEPOSITIONROW LINE13+3
        23,   // L_VIDVOLTAGEPOSITIONCOL
        0,    // L_VIDVOLTAGEPOSITIONDSPL
        12,   // L_AMPERAGEPOSITIONROW LINE15+10
        23,   // L_AMPERAGEPOSITIONCOL
        1,    // L_AMPERAGEPOSITIONDSPL
        12,   // L_PMETERSUMPOSITIONROW LINE15+16
        14,   // L_PMETERSUMPOSITIONCOL
        1,    // L_PMETERSUMPOSITIONDSPL
        11,   // L_CALLSIGNPOSITIONROW LINE14+10
        10,   // L_CALLSIGNPOSITIONCOL
        1,    // L_CALLSIGNPOSITIONDSPL
};

void checkEEPROM(void)
{
        // For H/W Settings
        uint8_t EEPROM_Loaded = EEPROM.read(0);
        if (!EEPROM_Loaded){
                for(uint8_t en=0;en<EEPROM_SETTINGS;en++){
                        if (EEPROM.read(en) != EEPROM_DEFAULT[en])
EEPROM.write(en,EEPROM_DEFAULT[en]);
                }
                // For items on screen.
                // First run, the default will be NTSC (show all data lines with NTSC systems that has only 13 lines)
                // In OSD menu' it's possible a quick default setup for PAL or NTSC
                for(uint16_t en=0;en<EEPROM_ITEM_LOCATION-EEPROM_SETTINGS;en++) {
                        if (EEPROM.read(en+EEPROM_SETTINGS+1) != EEPROM_NTSC_DEFAULT[en])
EEPROM.write(en+EEPROM_SETTINGS+1,EEPROM_NTSC_DEFAULT[en]);
                }
```

```
	}
}

void writeEEPROM(void)
{
	// For Settings
	for(uint16_t en=0;en<EEPROM_SETTINGS;en++){
		if (EEPROM.read(en) != Settings[en])
EEPROM.write(en,Settings[en]);
	}
	// For Position of items on screen
	for(uint16_t en=0;en<EEPROM_ITEM_LOCATION-
EEPROM_SETTINGS;en++){
		if (EEPROM.read(en+EEPROM_SETTINGS+1) !=
Settings[en+EEPROM_SETTINGS+1])
EEPROM.write(en+EEPROM_SETTINGS+1,Settings[en+EEPROM_S
ETTINGS+1]);
	}
}

void readEEPROM(void)
{
	// For Settings
	for(uint16_t en=0;en<EEPROM_SETTINGS;en++){
		Settings[en] = EEPROM.read(en);
	}
	// For Position of items on screen
	for(uint16_t en=0;en<EEPROM_ITEM_LOCATION-
EEPROM_SETTINGS;en++){
		Settings[en+EEPROM_SETTINGS+1] =
EEPROM.read(en+EEPROM_SETTINGS+1);
	}
}

// back to default setting & position for PAL/NTSC
void WriteScreenLayoutDefault(void)
{
	if (Settings[S_VIDEOSIGNALTYPE]){  // PAL
```

```
                for(uint16_t en=0;en<EEPROM_ITEM_LOCATION-
EEPROM_SETTINGS;en++) {
                        if
(EEPROM.read(en+EEPROM_SETTINGS+1)                      !=
EEPROM_PAL_DEFAULT[en])
EEPROM.write(en+EEPROM_SETTINGS+1,EEPROM_PAL_DEFAU
LT[en]);
                }
        }
        else {
                for(uint16_t en=0;en<EEPROM_ITEM_LOCATION-
EEPROM_SETTINGS;en++) {
                        if
(EEPROM.read(en+EEPROM_SETTINGS+1)                      !=
EEPROM_NTSC_DEFAULT[en])
EEPROM.write(en+EEPROM_SETTINGS+1,EEPROM_NTSC_DEFA
ULT[en]);
                }
        }
        readEEPROM();  // Refresh with default data
}
```

# BAB 5
# PENUTUP

Dari hasil yang telah didapatkan selama proses pembuatan *hardware* dan *software* untuk Tugas Akhir ini, maka dapat diambil beberapa kesimpulan dan saran untuk dapat dilakukan perbaikan dan pengembangan agar nantinya bisa lebih bermanfaat.

## 5.1 Kesimpulan

Berdasarkan data hasil diperoleh beberapa kesimpulan antara lain sebagai berikut:

1. Dalam pengembangan Minimosd ini menggunakan dua Minimosd untuk melengkapi beberapa menu sensor yang tidak tersedia di ArduPilot Mega.
2. Dilakukan *extra wiring* pada Minimosd untuk menambahkan menu-menu sensor yang akan ditampilkan pada layar monitor FPV.
3. Animasi untuk lengan robot tidak bisa digambarkan sesuai dengan bentuk real lengan robot oleh karena penggambaran animasi yang hanya dapat dilakukan dengan pengembangan 256 data karakter saja.
4. Resolusi antara layar yang dihasilkan oleh Minimosd dengan layar oleh kamera berbeda, sehingga hasil yang didapatkan banyak data karakter yang terpotong.

## 5.2 Saran

Terkait dengan kendala dan kekurangan dalam penyusunan Tugas Akhir ini, ada beberapa hal yang dapat penulis sarankan untuk pengembangan selanjutnya. Antara lain sebagai berikut:

1. Pengembangan OSD dengan beberapa fitur menu dirancang dengan bentuk hardware yang lebih minimalis agar tidak memakan tempat pada saat diaplikasikan pada robot atau kendaraan tanpa awak.
2. Melakukan pengembangan dengan hanya menggunakan satu buah Minimosd agar lebih efien dari segi ukuran barang dan lebih hemat dari segi biaya.

3. Membuat dan merancang sendiri sebuah produk hasil gabungan dari ArduPilot Mega dan Minimosd yang bisa dinamakan ArduOsd.

Demikian saran yang dapat penulis sampaikan. Semoga dapat bermanfaat untuk ke depannya.

# DAFTAR PUSTAKA


[1] Arduino Founder, Introduction of Arduino, diakses pada tanggal 1 Maret 2016, https://www.arduino.cc/en/Guide/Introduction.

[2] Atmel, ATmega328, diakses pada tanggal 1 Maret 2016, http://www.atmel.com/devices/atmega328.aspx.

[3] Marsono, Cara Kerja OSD (*On Screen Display*), diakses pada tanggal 12 Maret 2016, http://marsonotv.co..id/2011/10/cara-kerja-osd-on-screen-display.html.

[4] ArduPilot Dev Team, "*Minim OSD Quick Installation Guide"*, 2012

[5] Yuan Jian-long1, Zhang Dian-fu2, "*Design of video sighting instrument based on Max7456"*, China, 2010.

[6] Windestal, David, "*The FPV Starting Guide*", RCExplorer, 2 Juni 2013.

[7] Patrick Bertagna, "*How does a GPS tracking system work?*", www.eetimes.com, 2010.

[8] *LM35 Datasheet,* diakses pada tanggal 1 Mei 2016, http://www.electroschematics.com/6393/lm35-datasheet/.

[9] Ala-Paavola, Jaakko, "*Software interrupt based real time clock source code project for PIC microcontroller*", 23 Agustus 2007.

[10] "*Ultrasonic Sensors",* diakses pada tanggal 17 Mei 2016, https://www.ia.omron.com/products/category/sensors/ultrasonic-sensors/.

[11] Civil Aviation Authority, "*Small Unmanned Aircraft – First Person View (FPV) Flying*", UK, 2014.

[12] APM Copter, *Minim OSD Quick Installation Guide*, Department of Electrical and Computer Engineering, 2010.

[13] Dillon, A., Richardson, J. and McKnight, C. (1990b) The effect of display size and paragraph splitting on reading lengthy text from screen. Behaviour and Information Technology 9 215-227.

## DAFTAR RIWAYAT HIDUP

**Muhammad Saiful Hak.** dilahirkan di Probolinggo, pada tanggal 30 Agustus 1992 merupakan putra kedua dari tiga bersaudara pasangan Bapak Joko Marlis dan Ibu Tutik Sriwiningsih. Penulis menamatkan sekolah di SDN Sukabumi III tahun 2005. Kemudian masuk ke SMPN 5 Probolinggo, tamat tahun 2008, dan melanjutkan di SMAN 1 Probolinggo pada tahun 2011. Tahun 2011, penulis melanjutkan pendidikan di D3 Teknik Elektro FTI (Fakultas Teknologi Industri), ITS (Institut Teknologi Sepuluh Nopember) Surabaya dan tamat pada tahun 2014. Selanjutnya penulis mengambil pendidikan S1 program LintasJalur di bidang dan tempat yang sama yaitu Jurusan Teknik Elektro, FTI - ITS Surabaya pada pertengahan tahun 2014. Penulis memilih bidang studi Elektronika dan mengambil topik Tugas Akhir di Laboratorium Mikrokontroler dan Mikroprosesor.

E-mail : msaifulhak@gmail.com